SERVO MAGAZINE

July 2004

Walk This Way

STEP INTO THE WORLD OF ROBOTICS

KIMURA'S CHORUS: The League of Electronic Musical Urban Robots

U.S. $5.50 CANADA $9.25

07>

0 74470 58285 4

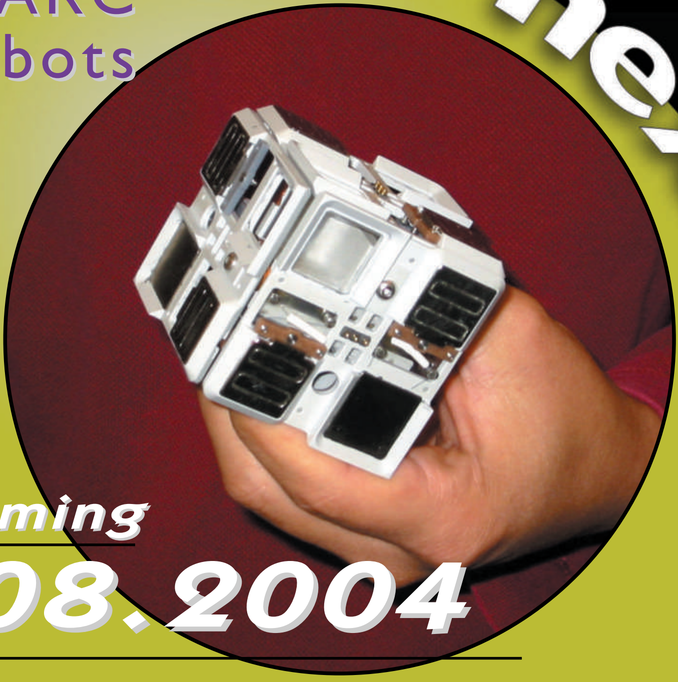
07.2004

Vol. 2 No. 7

SERVO table of contents

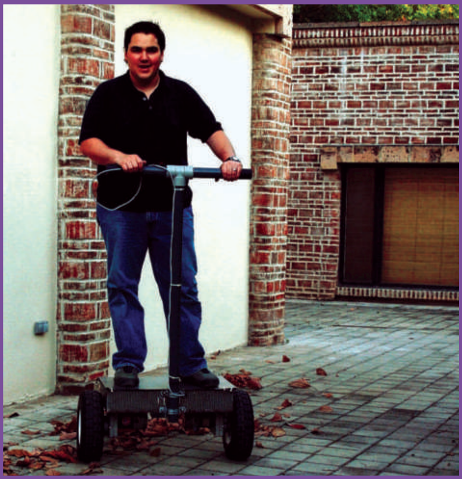
In our next issue

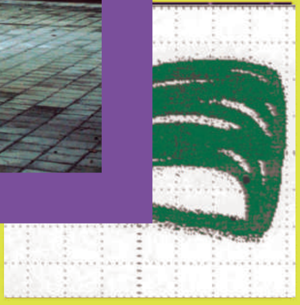
PARC Polybots



Coming 08.2004

SERVO Magazine (ISSN 1546-0592/CDN Pub Agree#40702530) is published monthly for $24.95 per year by T & L Publications, Inc., 430 Princeland Court, Corona, CA 92879. APPLICATION TO MAIL AT PERIODICALS POSTAGE RATE IS PENDING AT CORONA, CA AND AT ADDITIONAL ENTRY MAILING OFFICES. POSTMASTER: Send address changes to **SERVO Magazine, 430 Princeland Court, Corona, CA 92879-1300** or Station A, P.O. Box 54, Windsor ON N9A 6J5.
cpcreturns@servomagazine.com





columns

departments

features

Mind / Iron

by Dan Danknick

When I traveled to the East Coast last summer to interview Joe Jones at iRobot, I really wasn't sure what to expect (both of him, as well as his company). I mean, what does a dual-role (consumer and military) robotics R & D plant look like inside? I've worked in the cubicle farms of dotcoms and the clean room environments of DRAM fabs, but — hey now — this was *robotics!* In the end, it looked just like a typical software company, but with a lot more hand tools lying around on the tables. Of course, there were disassembled robots everywhere.

Later that year, I also had the chance to visit PARC (formerly Xerox's Palo Alto Research Center) and was once again curious about what I would see. The building was very high tech, but the content was the same: normal-looking offices, piles of electromechanical parts, and lots of robots in various states of development (or disassembly).

I've been noodling on a question for awhile now: What really separates the average, personal robotics tinkerer from the professional researcher? Is it four years of calculus, programming PIC micros in assembly, or even memorizing the pinouts of the 7400-series ICs? I've finally concluded that it's much simpler than that. People become great in what they do when they're willing to fail at their efforts.

Normally, we are exposed to great inventors and researchers only in the context of their current, crowning glory. "Here is Jonas Salk, creator of the vaccine for polio." I wonder how many dead ends Dr. Salk ran into over the eight-year span it took him to perfect that vaccine. Why didn't he give up after just seven years? I mean, let's be honest, most of us will give up on a project after only seven hours of discouraging results!

There are — by any measure — a lot of people interested in robotics these days. Attendance is increasing in local clubs and more complex toys are being sold to meet the intellectual demand of kids. Still, in any given club, the 90/10 rule still seems to apply: 10% of the members show-and-tell 90% of the robot technology. So, how do you transition from robo-observer to robo-researcher? Well, here is *Editor Dan's Road to Robotics List* (feel free to put it up on your wall):

- Tinker — Take things apart to see how they work. Then, try to reassemble them.
- Test — View everything as an experiment that is without a right or wrong outcome.
- Tell — Find someone you can work with. It will keep you motivated and expose you to new ideas.
- Teach — Convey your success to others — whether on the Internet or in the pages of *SERVO Magazine* — so they can benefit from it.

I'm pretty sure that Joe Jones didn't know the details of how he would architect a robotic vacuum when he started in at iRobot, but the fact that you can now buy one for about $200.00 shows that he didn't let a list of uncertainties slow him down.

My friend Mark used to remind me that the only bad mistake is the one you don't learn anything from. So, make a plan and try a new robotic project today! SV

Published Monthly By
The TechTrax Group — A Division Of
T & L Publications, Inc.
430 Princeland Court
Corona, CA 92879-1300
(951) 371-8497
FAX **(951) 371-3052**
www.servomagazine.com

Subscription Order ONLY Line
1-800-783-4624

PUBLISHER
Larry Lemieux
publisher@servomagazine.com

ASSOCIATE PUBLISHER/ VP OF SALES/MARKETING
Robin Lemieux
robin@servomagazine.com

ADVERTISING SALES DIRECTOR
Rich Collins
rich@servomagazine.com

MANAGING/TECHNICAL EDITOR
Dan Danknick
dan@servomagazine.com

ASSOCIATE EDITOR
Alexandra Lindstrom
alexa@servomagazine.com

CIRCULATION DIRECTOR
Mary Descaro
subscribe@servomagazine.com

WEB CONTENT/STORE
Michael Kaudze
michael@servomagazine.com

PRODUCTION/GRAPHICS
Shannon Lemieux

COVER DESIGN
Michele Durant

STAFF
Janessa Emond
Kristan Rutz

OUR PET ROBOTS
Guido
Mifune

BIO-FEEDBACK

Dear *SERVO*,

In the "Geerhead: Medieval Automatons" article, there was a small error: Strassbourg is in France, not in Germany. Okay, they speak German there, but it really is a French city. Strassbourg is the capital of the Alsace region, which the French "inherited" after WW-I.

Jan Verhoeven
via Internet

We have a technical loophole in that it still belonged to the Germans during the period we are talking about, so it was in Germany then.

— David Geer, Columnist

Dear *SERVO*,

Regarding Tom Carroll's article in the April issue, where are Elektro and Sparko now? Someone within Westinghouse must have some idea where they are today.

Richard Tomkins
via Internet

Dear Richard,

Your Email was forwarded to me for a reply. We do not exactly know what happened to Elektro and Sparko. I do know that Elektro was brought out of mothballs and put back onstage with Sparko here in Pittsburgh, PA, in the year 1949, but .... after that ... we just don't know — fascinating characters, Electro and Sparko!

Ed Reis
Executive Director
George Westinghouse Museum

Leave it to SERVO to find Elektro and Sparko! In the coming issues, you will learn what these robots have been up to.

Dear *SERVO*,

I've really enjoyed the article series by John Myszkowski and his passing is a great loss to the hobby robotics community.

Mark Weston
via Internet

got bot?

Whether you have a build, code, or theory to share, *SERVO* wants to know what you — the resident of the robot workshop — are creating. We want you to Email us your article submissions. Some topics of interest are:

- Sensors and signal processing
- Mechanical fabrication
- Software techniques
- Data protocols
- Unique drive geometries
- Material selection and use
- Distributed communication

In Loving Memory ...

We are saddened to report the passing of John Myszkowski, our *Cutting Edge Robotics* project writer. John came onboard starting with the January issue and put a remarkable amount of effort and creativity into his work. We always looked forward to his next C. E. installment and never really expected it to end. One of his coworkers — a fellow roboteer — hopes to complete John's goals with the C. E. series in the next few months.

A resident of Ontario, Canada, John is survived by his wife and two children. He will be deeply missed by the staff of *SERVO* and its readers.

Dear *SERVO* readers,

There is an error in my article, "Bot Builder + Couch Potato," in the May issue of *SERVO Magazine*. On page 61, in the section headed "Programming the RCA Universal Remote," in the second paragraph after the heading, the code to type in is 1002 not 1001.

Bob Knoblauch

Dear *SERVO*,

I liked your May 2004 "Mind/Iron" editorial, but I think there is a dangerous line between "basic English" and "sales English." Selling a product is obviously an important job and many engineers do well at it, but over-hyping things can get you into trouble. If things are oversold, there's too much expectation and, when the miracles don't happen, the basics are forgotten. I think that's what happened with the DARPA Grand Challenge and it's important for both DARPA and the companies involved to explain things in a basic way. Too much hype is just as bad as too much detail.

Keep up the good work with *SERVO*. Maybe one of these days my kids will force me to buy a real robot!

Mike Rosing
via Internet

Announcing Our New Area Code

As of July 17, our area code will change from (909) to (951). This will affect both our phone and fax numbers.


TETSUJIN 2004
October 21-23
Santa Clara, CA
Where Man Incorporates Machine
In Conjunction With:
RoboNexus
The International Business Development, Educational and Consumer Event For Personal, Service and Mobile Robotics
Santa Clara Convention Center, October 21-23, 2004
Although the June 14th registration deadline has passed, there are still a few spots open for serious teams to enter this revolutionary robotic event!
Don't let this opportunity pass you by — sign up to participate in SERVO Magazine's powered exoskeleton weightlifting competition!
www.servomagazine.com

ARES DUCT CRAWLER

BY MICHAEL SIMPSON

Out of curiosity, I purchased one of those mold test kits from my local home center. With this type of kit, you expose a small dish to your air ducts and wait 72 hours for the results. If mold appears in the dish, you need to send the dish and $30.00 to a lab to obtain professional results. I patiently waited the 72 hours and was surprised when it indicated I had a serious mold problem. I was not convinced I had a problem, so I decided to investigate further.

I had just purchased a small, 2.4 GHz camera from a mail order house. The camera transmits to a small receiver that you hook up to a TV or VCR. I tried to place the camera inside one of my ducts, but, since I could only see a few feet, it did not help. I decided to attach the camera to a small bot and see if I could move it further into the duct. The bot had to fit into the 4" x 10" hole seen in Figure 2. Once inside, there was plenty of room to maneuver.

I had tried to use a 75 MHz R/C radio, but could not get it to work inside the metal ductwork. I decided that the use of IR was perfect for this situation; I imagined that the metal ductwork would actually allow the IR beams to bounce around and extend the range.

I decided on the Ares Robot controller board because it has a coprocessor that handles all the motor and servo processing. The main processor on the Ares board is the Dios, which uses a VB-type language, so programming the bot was a snap. I used four motors to drive and steer the bot and a servo to control the angle of the servo. The Ares board also supports both dual and single power systems. I used a six cell battery pack as a single power source for both the motors and logic.

My First Attempt

The four wheel differential (skid steer) system worked great. The motors on each side of the bot were connected in parallel, then connected to one of two motor channels on the Ares board (Figure 1).

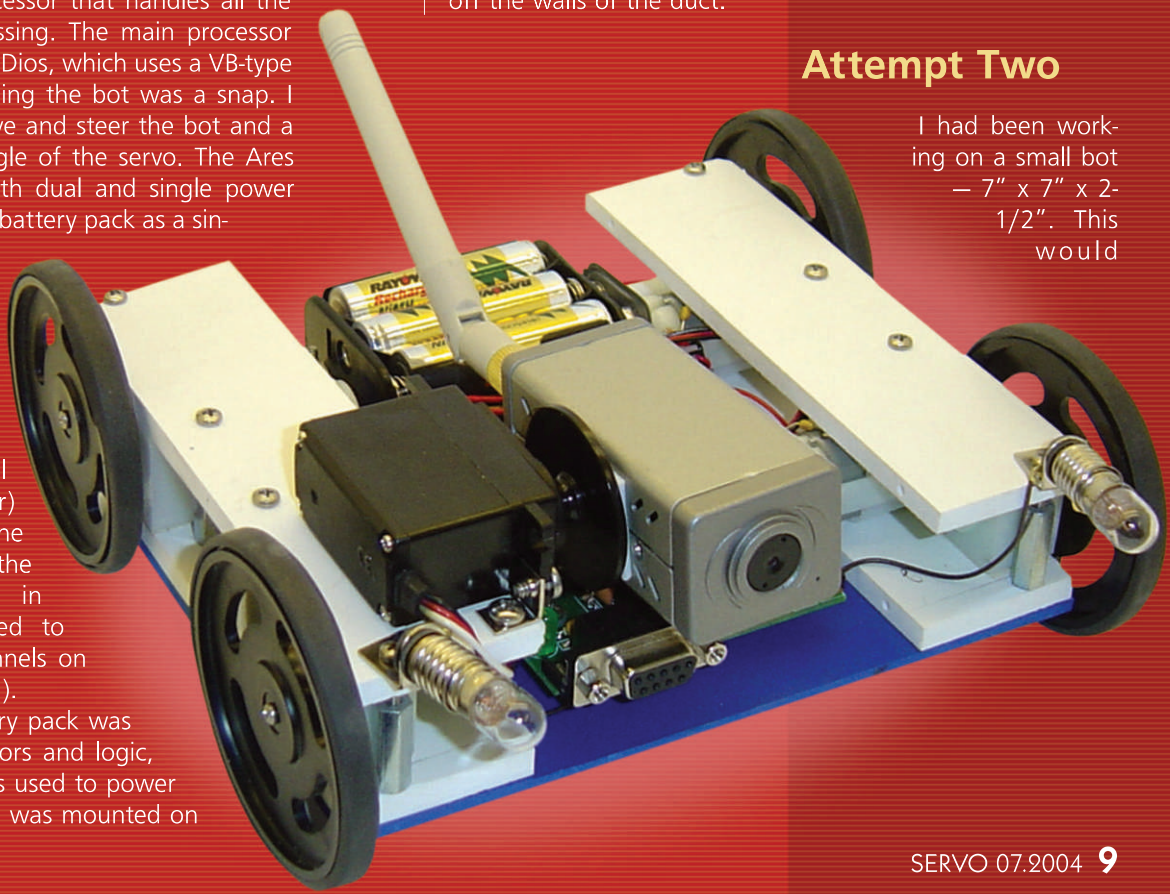
A six cell, AA battery pack was used to power the motors and logic, and a 9 volt battery was used to power the camera. The camera was mounted on a small pivot at the front of the bot and its up and down movement was controlled by a small wire connected to a standard servo. The batteries were located on the underside of the bot and were held in place with Velcro® straps. I used a small, LED flash light on the top of the camera to help illuminate the ducts.

The first attempt worked fairly well; however, I did encounter a few problems that made it clear that a redesign was needed. The small, white LED penlight did not provide enough light to illuminate the ducts for the camera. It was difficult just to see enough to move the bot, so inspecting the duct walls was out of the question. As I moved the bot down the duct, I encountered a duct flow valve (Figure 2). through various legs of the duct work.

It was obvious that the bot needed to have a lower profile so that it could slide under the valve when it was in the open position. It was important that I make it past this control valve because I wanted to inspect the main plenum. I originally used a universal TV remote to transmit movement codes to the bot. I had to hold the transmitter in the vent to use it. As the bot rounded the corner in one of my ducts, I lost control. So much for my idea that the IR would bounce off the walls of the duct.

Attempt Two

I had been working on a small bot — 7" x 7" x 2-1/2". This would



make a great bot for the duct crawler. I made the main base out of 1/8" and 1/4" expanded PVC. Expanded PVC is great to work with because you can tool it with normal wood working tools. You can even score it with a razor knife and break it in a pinch. Expanded PVC can usually be purchased at a local sign shop.

In my case, the sign shop just gave me their scraps. If you don't have access to expanded PVC, you can also use fiber board, wood, or Plexiglass.

In the second attempt, I used two small gear motors, connected in parallel on each side of the bot. In order to make the bot as compact as possible, I built two motor assemblies, as shown in Figure 3.

The motor assemblies were just two pieces of 5-1/2" x 1-1/2" x 1/4" expanded PVC with two of the gear motors sandwiched in between them. I used double-sided carpet tape to hold the gear motors in place while I positioned the PVC. Everything is really held in place by 3/4" standoffs and #4 machine screws. Figure 4 shows how each of the motor assemblies was mounted on one side of a 5-1/2" x 6-1/2" x 1/8" piece of expanded PVC.

Figure 5 shows how I attached all of the wires to a female header so they could be plugged into the Ares board. I used small, 1/4" standoffs for mounting the Ares. The other standoffs in that figure are used to hold the main six cell battery in place.

Figure 6 shows the Ares board mounted securely to the standoffs, and nestled between the "drive pods".

The camera was attached directly to a round horn connected to a servo mounted on top of one of the motor assemblies. The servo is controlled by the Ares board and allows me to move the camera up or down.

By spinning the bot, I can view just about any angle. For lighting, I attached two 6 volt incandescent flash light bulbs directly to the switched Vin on the Ares board. These bulbs mounted on two holders, which can both be purchased from RadioShack.

One thing I discovered with the first attempt was that the 9 volt battery would run out of power very quickly when connected to the camera. I could not take the chance of this happening while the bot was deep inside the duct work, so I used a larger, 9.6 volt NiCad battery pack on this attempt. Figure 8 shows the red battery pack strapped to one side of the bot.

The Ares board has a provision for a small IR module. This makes hook-up very simple. Complete directions on attaching the components discussed in this article can be found in the Ares manual.

Dios Operational Program

The Dios program is straightforward. It makes a call to the built-in IRread function and branches to a section in the code to handle the particular switch.

Each switch has three states, so a branch command is used to jump to the various handler code for that switch position.

Transmitter Commands handled:

Switch 1: Control Motor 1 Forward/Stop/Reverse
Switch 2: Control Motor 2 Forward/Stop/Reverse
Switch 3: Control Bot Speed
Switch 4: Control Camera Angle

```
func main()
   pause 1
   dim cmd
   dim camerapos
   dim botpower

   print "reset"
   CPinit()
   CPMotorReset(0)

   botpower = 190
   gosub setpower
   camerapos = 7500
   CPAres_setservo(3,camerapos)
   CPMotorDampervalue(0,100)

loop:
```

FIGURE 1 — The version 1 duct crawler.

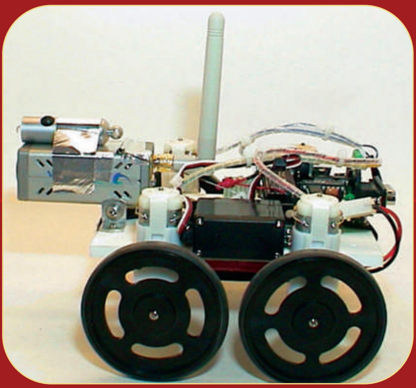


FIGURE 2 — The duct flow valve.

FIGURE 3 — The motor modules.

```
    cmd = IRread(0,10000)
    if IRdevice = 0 then
         goto loop
    endif

'print IRcmd," ",IRdevice
    branch IRdevice,loop,dosw1,dosw2,dosw3,dosw4
    goto loop

'———-Switch 1 Left Motor Forward or Reverse ————
dosw1:
    branch IRcmd,sw1cmd0,sw1cmd1,sw1cmd2

sw1cmd0:
    CPMotor2stop(0)
    goto loop

sw1cmd1:
    CPMotor2fwd(0)
    goto loop

sw1cmd2:
    CPMotor2rev(0)
    goto loop

'———-Switch 2 Right Motor Forward or Reverse ————
dosw2:
    branch IRcmd,sw2cmd0,sw2cmd1,sw2cmd2

sw2cmd0:
    CPMotor1stop(0)
    goto loop

sw2cmd1:
    CPMotor1fwd(0)
    goto loop


sw2cmd2:
    CPMotor1rev(0)
    goto loop

'
 ———Switch 3 Bot Speed —————————-
dosw3:
    branch IRcmd,sw3cmd0,sw3cmd1,sw3cmd2

sw3cmd0:
    goto loop

sw3cmd2:
    botpower=botpower - 10
    print botpower
    if botpower < 140 then
       botpower = 140
    endif
    gosub setpower
    goto loop

sw3cmd1:
    botpower=botpower + 10
    print botpower
    if botpower > 255 then
       botpower = 255
    endif
    gosub setpower
    goto loop

'———Switch 4 Camera Angle —————
dosw4:
    branch IRcmd,sw4cmd0,sw4cmd1,sw4cmd2

sw4cmd0:
    goto loop

sw4cmd1:
    camerapos=camerapos - 100
    print camerapos
    if camerapos < 7000 then
      camerapos = 7000
    endif
    CPAres_setservo(3,camerapos)
    goto loop

sw4cmd2:
    camerapos=camerapos + 100
    if camerapos > 11000 then
       camerapos = 11000
    endif
    CPAres_setservo(3,camerapos)
    goto loop


setpower:
    CPMotor1speed(0,botpower)  ' Right Motor
    CPMotor2speed(0,botpower)  ' Left Motor
    return

endfunc
```

FIGURE 4 — Mounting and wiring the motor modules.

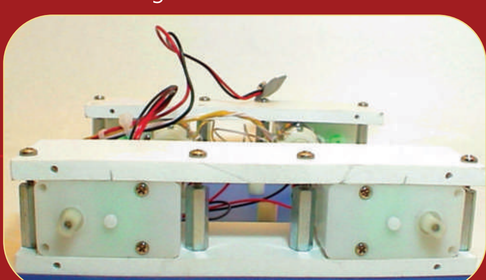


FIGURE 5 — Standoffs on the blue PVC base.

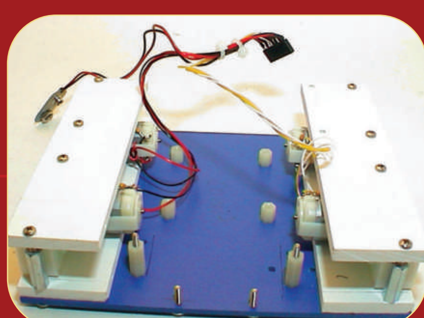
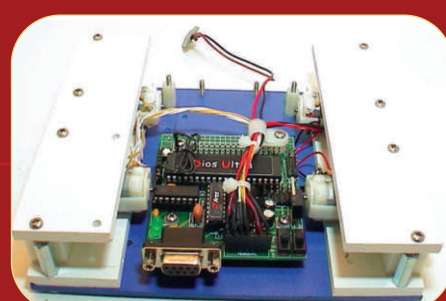


FIGURE 6 — The Dios control board is mounted.



```
include \lib\DiosCPMotor.lib
include \lib\DiosIR.lib
include \lib\DiosCPAres.lib
```

Controller Transmitter

Originally, I used a universal TV remote. For the second bot, I decided to completely redesign the remote control. I used one of the Kronos Robotics motor switch boxes, as shown in Figure 9. These control boxes have four, three-state switches.

I used the Athena microcontroller to create an interface to the switch and to send the IR data. The Athena has built-in IR transmitter and receiver commands. All you have to do is connect an IR LED to port 6 and you can start transmitting IR codes.

I used the Athena Carrier 1 board for the IR transmitter with a couple of headers. The board is mounted on the four AA cell battery holder (Figure 10). If you lose control of the bot, simply poke a small, 1/4" small hole in the duct and insert the transmit LED to regain control.

A 2N2222 transistor is used to drive the IR LED for more power. If you use a transistor other than the one listed, you may need to place a 1K-10K resistor between the I/O port 6 of the Athena and the base of the transistor.

Controller Program

The Athena program monitors the four switches on the motor switch. Every 25 ms, a code is sent to the duct bot with instructions on the state of each switch, as shown in the following program:

```
'motor switch controller for duct bot
   dim num,stat1,stat2,stat3,stat4
   clearall
   pullupon

   dim swno

loop:
   swno=1 : gosub check1
   swno=2 : gosub check2
   swno=3 : gosub check3
   swno=4 : gosub check4
   goto loop
```

FIGURE 7 — Front of the decked out crawler.

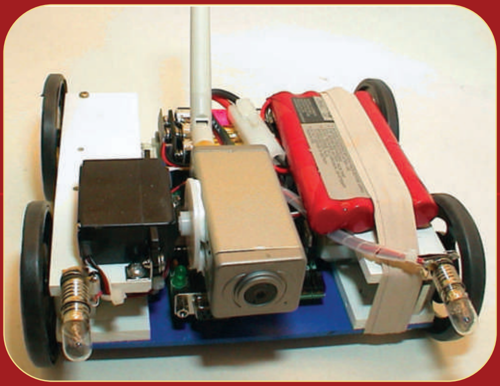


FIGURE 8 — Notice the large Tx antenna for the camera.

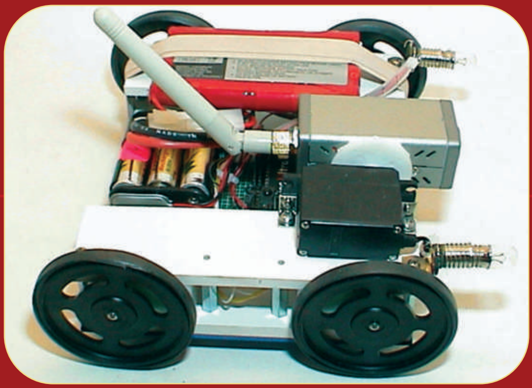
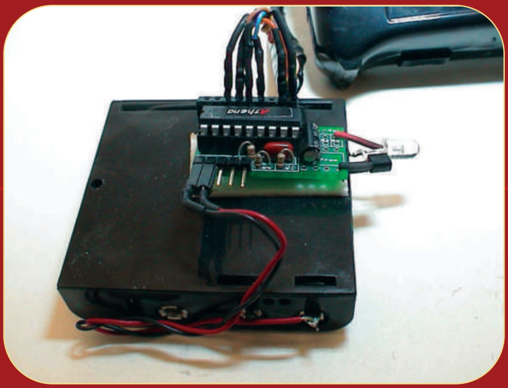


FIGURE 9 — The robot control box.

FIGURE 10 — The Athena carrier on the battery pack.



```
check1:
   configio 4
   setio
   goto cont

check2:
   configio 5
   setio
   goto cont

check3:
   configio 7
   setio
   goto cont

check4:
   configio 8
   setio
   goto cont

cont:
   portbitget num,0,2
   portbitget num,1,3
   if num = 3 then
      num = 0
   endif

   configio 8
   irout num,swno
   pause 25
   return
```

Final Thoughts

Using the duct crawler, I was able to determine that I had no visible mold in my duct work or furnace; I consulted an expert. I found out that many of the mold kits are rackets. They show mold and expect you to send in the results (plus $30.00) for an exact assessment. The expert advised me that, if you cannot see or smell a mold infestation, chances are that you do not have one.

If you decide to build this bot or a similar one, please feel free to experiment. I'm sure there are many enhancements that can be made. You are not restricted to using motors, as the Ares can also control full rotation modified servos. **SV**

SOURCES

Most of the parts for this bot can be found on the Kronos Robotics website at **www.kronosrobotics.com**

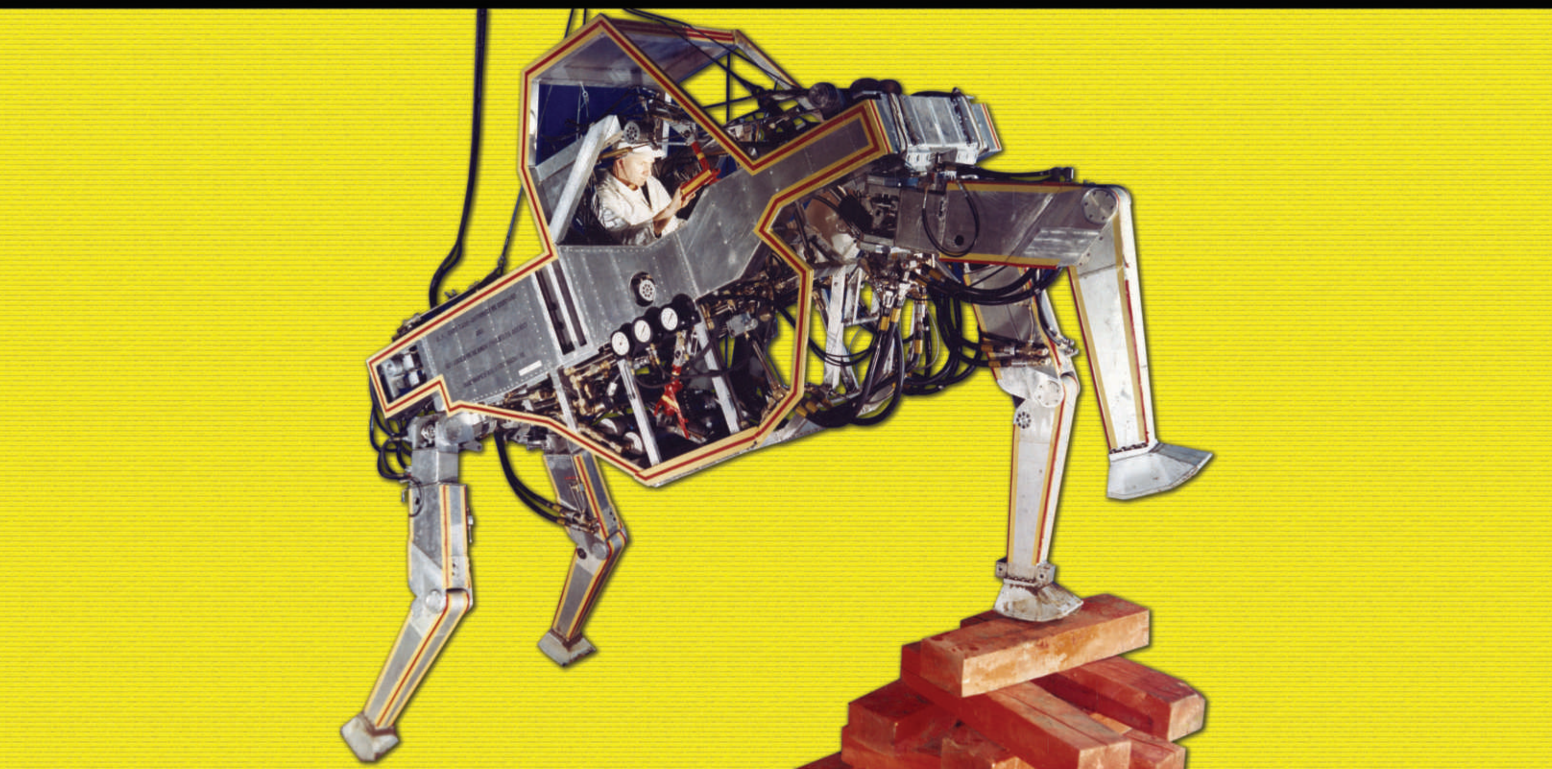
Ares Duct Crawler
Ares robot board #16328
Gear motor/wheel/capacitor kit #16377
IR module #16226
Six AA cell battery holder #16321
9 volt battery vlip (for six cell pack) #16264
Standard servo (any will do) #16317

Controller Transmitter
Athena #16276
Athena Carrier 1 #300
Motor switch #16247
Four AA cell battery holder #16323
IR LED #16223
2N2222 transistor #16143

Wireless Camera
Circuit Specialists #GFP-2400

GEERHEAD

by David Geer
geercom@alltel.net



Where Have All the Robots Gone?

To warehouses, scrap heaps, fond memories, and books that are out of print! (Which doesn't necessarily make them unrecoverable!)

Remember When ...?

Remember Todd Loofbourrow and the Keyboard Input Monitor (KIM-1) robot? How about Frank DaCosta's robot pet? Soon, you will remember them, as well as the GE Walking Truck, the Hardiman Suit, and the Hughes Aircraft Mobots, as well as the works of David L. Heiserman and Edward L. Safford, Jr. Let's begin.

Todd Loofbourrow and the KIM-1 Robot

Construction

Microtron was made of plywood, sheet metal, and angle aluminum. The final creative effort after several stages — including building the framework and empowering the bot with sensors — is speech recognition capabilities. (It's all in his book. Keep reading and see methods for finding old robotics books in one of the sidebars.)

Capabilities

Microtron could carry up to a whopping 600-odd lbs or push up to 150 lbs, depending on the task at hand. How many home bots are that tough today?

Microtron could be manipulated via joystick or could wander off all on its own. A true, self-directed robot, it could cut a predetermined path across the floor with a simple switch from joystick to self-control mechanism.

The Last We Heard

Loofbourrow's intentions were to evolve his KIM-1 robot, Microtron, by adding a computer-generated voice, real working arms, and sensing eyes.

The Book

Author of *How to Build a Computer*

Our readers make the difference! A special thanks goes out to roboticist and regular SERVO Magazine reader Andrew L. Ayers of Glendale, AZ for suggesting this topic.

Controlled Robot, Loofbourrow was fascinated with the creation of artificial intelligence. Loofbourrow's KIM-1 controlled robot — "Microtron" described here — was built and then rebuilt in words for the interested onlooker, chapter by chapter in this book.

If you're going book hunting, here's what you'll be looking for. *How to Build a Computer Controlled Robot*, by Todd Loofbourrow, published by Hayden Books in 1978, ISBN 0-8104-5681-8.

BTW, What's a KIM-1?

KIM-1 was developed by MOS, an IC factory bought by Commodore. KIM-1 computers had 2,048 bytes of ROM and 1,152 bytes of RAM. The computer was created for use with teletyper/teleprinters. A KIM-1 can house programs on papertape (hardly available today) or via cassette recorder.

David L. Heiserman

Robots

In four books, Heiserman demonstrates Evolutionary Adaptive Machine Intelligence (EAMI), robots that learn from their environments by building gradually more complicated bots that are more and more intelligent.

The bot projects — from simplest to most complex — instruct the reader on how to build Heiserman's robots Buster, Rodney, his virtual robot systems, and finally the Parabots.

RB5X Gets a Brain

Heiserman is responsible for the Alpha and Beta self-learning software in the popular consumer robot, RB5X, which empowered the robot to absorb and employ information from its surroundings.

With Heiserman's software, the RB5X starts with random responses and builds the capability to make assumptions about its environment. The robot could learn from its mistakes, such as things it bumped into; eventually, it would stop running into them.

Books by Heiserman — All From TAB Books

Build Your Own Working Robot, ISBN 0-8306-1181-9 (for the hard copy copy), published April 1976. *How to Build Your Own Self-Programming Robot*, ISBN 0-8306-9760-8 (hard copy) and 0-8306-1241-6 (paper), published September, 1979.

Other Heiserman books include *Robot Intelligence* (with experiments), ISBN 0-8306-9685-7 (hard copy) and 0-8306-1191-6 (paperback), published in January 1981. Finally, *How to Design and Build Your Own Custom Robot*,

Finding Old Robotics Books

Small towns like the one I grew up in are great places to find old robot books. Why? First, judging by my experience, they are usually 20 years behind the times and, second, small town organizations and communities are less likely to throw things out (in part because they can't afford to replace them).

So, 20-, 30-, and even 50-year-old robotics books may still be in use in the public libraries, high school and junior high libraries, and vocational school and even junior college libraries.

Your local library — in a big or small town — likely has something comparable to an InterLibrary Loan program. You may be asked to fill out a form and pay a small fee for delivery to your local library if the book is found, but searches can span most every library in the country. When your book arrives, you generally have the typical two weeks to read and enjoy.

Surprisingly, I was able to find many old robotics books for sale by using the following resources — **www.biblio.com/index.php** or **www.bookfinder4u.com/**

ISBN 0-8306-9629-6, published in 1981.

Frank DaCosta

Robots

Frank DaCosta's book presents the most obedient pet you'll likely ever find — a robotic one ("all assembly required," but at least you'll never need a pooper-scooper).

Frankly, it doesn't really resemble most critters I've ever seen, but you might call it a mutt for all its hybrid parts. The microcontroller is assembled around an 8085A eight-bit microprocessor.

It's All in the Details

Thankfully, the book goes into great detail about the construction of your new pet from, uh, head, to, er, toe (including hardware, software, expansion modules for language, hearing and navigation, so to speak — get the book).

Many problems in the book can be cast in today's technology for a more updated solution and, thereby, a present-day rendition. (It still may not look much like Fido, but, hey, what were you expecting? If you weren't more at home with robots than cats and dogs, I'm not sure you'd have read this far, anyway.)

What's Cool Here Is ...

The SoniScan object avoidance technology, the Excom technology used to communicate with the robot, and the Audigen technology that lets Fido or whatever talk back are the coolest. Key to emulating behaviors comparable to those of a house pet is the Artificially Random Self-Motivation software (ARASEM).

DaCosta Book Rocks!

This truly detailed guide takes you from start to finish in building your own robotic pet and programming the little rabble-rouser, as well. Look for *How to Build Your Own Working Robot Pet*, ISBN 0-8306-9796-9 (hard cover) and 0-8306-1141-X (paperback), from TAB Books, published in June, 1979.

Edward L. Safford, Jr.

Safford's *The Complete Handbook of Robotics* brings us information and representations of the Hughes Aircraft Mobots, the GE Walking Truck, and the Hardiman Suit.

"Walk Like a Truck"

The Walking Truck finally afforded us mobility like that of our human legs — mobility for terrains that wheels couldn't surmount. The Walking Truck was a

Newer doesn't always mean better. Can your robot hulla hoop?

METAL MUSCLES

A literal-minded approach to body-building

A CONTRACT has been awarded the General Electric Company for development and construction of a set of "mechanical muscles" that will give an ordinary man the strength of a giant.

By means of an advanced system of levers, control linkages, and servomechanisms, this unique machine will mimic and amplify the movements of its operator — dramatically extending his strength and endurance. This man-machine marriage will essentially combine the operator's dexterity, brain-power, and all-round versatility with a machine's strength, size, and ruggedness.

A research prototype of the "mechanical muscles" machine, now being developed at the Research and Development Center, is scheduled for delivery in early 1968. The project is funded under a program jointly supported by the U.S. Army Natick (Massachusetts) Laboratory and the U.S. Office of Naval Research.

Worn like an external skeleton, the "mechanical muscles" machine — nicknamed "HardiMan" — will permit its operator to lift a 1500-pound load while exerting only a fraction of this force. He will be able to perform general load-handling tasks, including walking, lifting, climbing, pushing, and pulling. The machine — technically described as a "powered exoskeleton" by its developers —will be attached to the operator at the feet, forearms, and waist.

Potential applications for the HardiMan are foreseen in warehouse and factory operations, bomb loading, and underwater salvage. Although the prototype will be connected to a separate power supply by means of flexible hydraulic lines, it is anticipated that later models will have self-contained power units.

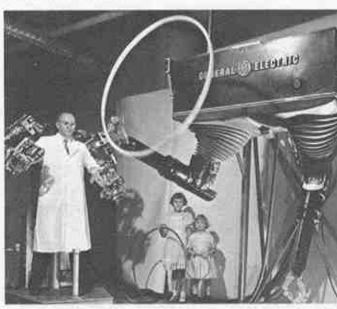
The contract resulted from 15 years of work on cybernetic anthropomorphous machines (CAMS) at the Research and Development Center. The engineering technology for "force feedback" control—the key to HardiMan — was developed by Ralph S. Mosher, who will guide development of the machine.

"Force feedback means that proportions of the forces generated or encountered by the machine are duplicated and reflected to the operator," Stanford Neal, manager of the Center's Mechanical Technology Laboratory, stated. "If the machine's arm or leg strikes a solid object, the operator feels that identical force situation of striking a solid object with his arm or leg."

"As a result, the machine simply becomes an extension of the man, and the operator is able to concern himself solely with performing the task at hand. Thus, man now has the ability to control a multi-motion machine in a natural way and to move loads at higher speed, with greater dexterity, than ever before. The control concept makes training time almost non-existent," Neal said.

The effectiveness of a man-machine marriage was first demonstrated by General Electric engineers during the 1950's with the construction of a remote-controlled manipulator with two claw-like hands. Forces encountered by the "slave" hands were fed back to the operator, enabling him to "feel" what he was doing. As a result, he could handle dangerous substances — such as radioactive materials — from a safe distance.

The HardiMan machine will be controlled by hydromechanical servovalves—the mainstays of the feedback system. •



PREDECESSOR: Manipulator that could simulate the clutching motion of the human hand is shown twirling a hoola hoop in this 1959 photo. Device is being operated by project engineer Ralph Moser, who is now playing a key role in the development of HardiMan.

On the next three pages, a look at the Mechanical Technology Laboratory.

GENERAL ELECTRIC R & D JOURNAL 9

Resources

Manuals, including KIM-1.
http://highgate.comm.sfu.ca/~rcini/classiccmp/my_docs.htm

Tracking down KIM-1 users.
www.ping.be/kim-1__6502

http://members.cox.net/oldcomp/kim1.htm

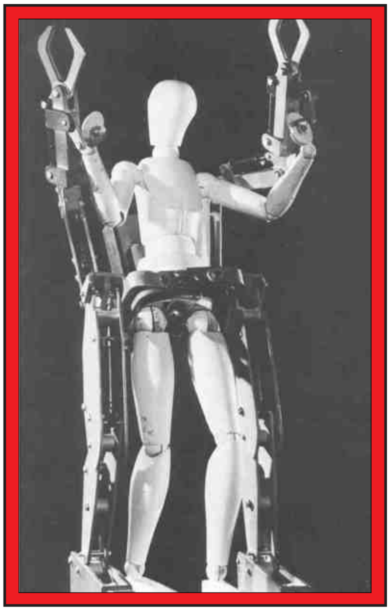
The RB5X.
www.robotswanted.com/robotgallery/rb5x/

History of Robots.
www.cs.bham.ac.uk/research/robotics/cbbc/history.php

large, 3,000 lb, crane-sized hydraulic vehicle created first for military and then industrial use.

Obviously made for heavy or difficult rather than quick tasks, the strong and well-balanced behemoth could do the job, but packed only a four mile per hour top speed.

The Walking Truck had a computer brain that assisted the manipulation of its legs, which were ultimately directed by its driver's hands and feet. Developed by Ralph Moser for GE in 1969, this metal workhorse's leg technologies achieved coordination that approached that of living four legged creatures for the first time.



The Hardiman Suit — Tetsujin, anyone?

Hughes Aircraft Mobots

In 1969, Hughes Aircraft created the Mobots or mobile robots — remote controlled machines for tasks that fell outside the realm of reasonable expectations placed on human beings. This included jobs in unbearable environments or for which people were not capable of. The workload of the Mobots included chemical testing, construction, and interacting with nuclear reactors.

The Hardiman Suit

Earlier in 1965, GE built the first nearly working (though impractical) exoskeleton. Unlike cyborgs where the robot is built into the human being, the exoskeleton is a removable or wearable outer robot. This robot was the Hardiman 1.

Weighing more than the average car of the day and being a room-full in size, the Hardiman was intended to lift 250 lbs while putting only 10 lbs of resistance onto the individual wearing it. This contraption was never fully operational and no one risked their life by attempting to wear or use it. Fueling it was also an obstacle that couldn't be surpassed.

The Books

Two books from Safford are presented here for your location and edification. *The Complete Handbook of Robotics*, ISBN 0-8306-9872-8 (hard copy) and 0-8306-1071-5 (paperback), published by TAB, Novembers 1978. *Handbook of Advanced Robotics*, ISBN 0-8306-2521-6 (hard copy) and 0-8306-1421-4 (paperback), also from TAB, published in 1982.

In addition to Safford's *Complete Handbook*, *The Robot Book* (Robert Malone) contains three large, clear images of the Hardiman, Mobots, and the Walking Truck. **SV**

Finding Old Robots

In addition to the links below, I have devised a plan for finding old robots. Once you locate the name of the publisher of old robotics books, look them up. They generally have someone in charge of keeping track of where their authors — past and present — are today. If you're lucky, they'll let you pass messages and requests to the author, who is often also the builder of many of these esteemed, historic robots.

http://web.mit.edu/museum/collections/science.html

http://robothut.robotnut.com/

www-robotics.usc.edu/old_robots.html

www.robotswanted.com/

http://members.tripod.com/bobgreiner/id29.htm

A Synthetic Nervous System for Robotic Locomotion

by Tom Jenner

Robotic nervous systems attempt to simulate the movement patterns of animals. With the exception of a few lower invertebrates, animals have a nervous system that utilizes central pattern generators to coordinate and synchronize their movements. The central pattern generator has a pacemaker neuron.

The pacemaker neuron, when combined with a phase-shifting network or interacting pacemaker neurons, causes the generation of an oscillating signal that is received at the muscle tissue through inter-neurons and motor neurons. These neurons communicate with voltage spikes and so this type of processing is called "spike based computing" (also known as "integrate and fire").

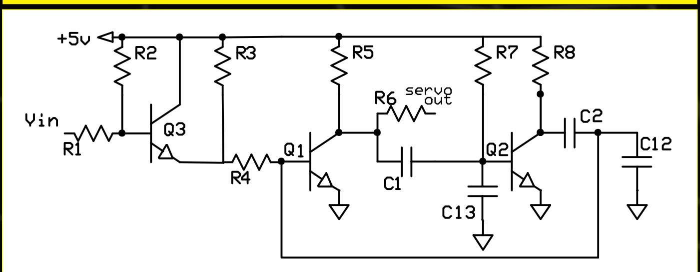
A spike based computing form of communication can be effective and robust, especially in a noisy environment where signal attenuation may occur over a long distance; e.g., from the spinal cord to the hand. Spike computing is also efficient at processing information in the time domain.

A substantial amount of research has taken place in the field of spike based computing with respect to robotics and artificial life. This research tends to be not only complicated, but expensive. Very complex circuits using custom VLSI (Very Large Scale Integrated) analog silicon, digital signal processors, or a combination of both have been created to simulate how a biological central pattern generator and nervous system work.

Others have attempted to create simple nervous systems for robots. Most notably, Dr. Mark Tilden — the founder of BEAM robots — uses an adaptive, oscillating ring network to pattern the movement of robotic legs, each of which is independent. It uses a pulse delay circuit wired up to another pulse delay circuit that functions as an artificial neuron.

Some of the disadvantages of this approach are that the motors have no idea where they are in their phase space (unless potentiometers are used or circuits that are set up as "edge cells," such as limit switches) and that they are not taking advantage of the computing power of variable analog circuitry. It should be understood, however, that Dr. Tilden was trying to use the simplest circuits possible to create a nervous system using the "integrate and fire" approach of biological neurons and was a pioneer in the field of robotics in the process.

Circuit 1. Basic motor neuron.



Nature Doesn't Have the Benefit of Silicon ...

This nervous system for locomotion differs from most (all?) others in that it can synthesize the action of biological nervous systems by using a wide variety of oscillators that can operate from around 1/2 to 3 Hertz. Also, there are

many ways to implement the other circuits shown.

The central pattern generator can be as simple as a single transistor sine wave oscillator with the output connected to a circuit (Circuit 1) that acts as a voltage to position converter using a few components and an unmodified, hobby type servo. When correctly connected together, the shaft of the servo will rotate back and forth in a smooth, near sine wave pattern. This back and forth motion — when properly in phase with other sine wave/servo combinations — forms the basis of robotic locomotion, whether it is walking, crawling, swimming, or flapping.

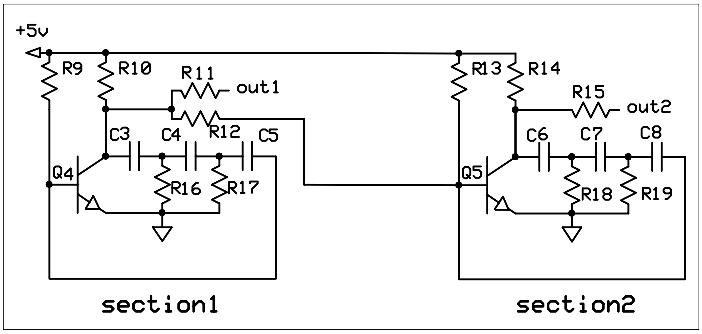
The nervous system can be used to synthesize all forms of limbed and finned robotic locomotion within the limits of current servo technology, provided that the circuits can generate the proper waveforms. Furthermore, only by modulating the frequency, phase, amplitude, and DC voltage offset of the oscillator can the full power of continuously variable analog computation be taken advantage of. In these examples, I use only NPN transistors to perform this function to keep things easy.

How It Works

Circuit 1 shows a circuit that I call the "basic motor neuron." It's a two transistor multivibrator (Q1, Q2) with a third transistor added (Q3) and a high impedance on its base that is functionally a voltage variable resistor (with a threshold). This outputs the 1-2 msec pulse that is needed to control the servos. The MPF102 JFET was tried as the third transistor, but I get much better and more linear results just using an inexpensive 2N2222 or something close to it. Other circuits have been tried using opamps and diodes, 555 timers, and just digitizing the signal, but this is the circuit I settled on, in keeping with the BEAM spirit.

Use the values suggested as a starting point; there's a lot of room for experimentation. Be sure not to omit C12 and C13, as these capacitors help eliminate noise in the oscillator. An oscilloscope will be helpful to adjust these circuits. R3 is a potentiometer that, with a voltage at R1, adjusts the high level pulse. R1 and R2 control the impedance of the circuits and determine how much an input voltage influences the timing. This circuit with a hobby servo represents a voltage to position converter.

Circuit 2 represents a central pattern generator using phase coupled sine wave oscillators. Section 1 eventually feeds to one servo and Section 2 feeds to another. The sine voltage at OUT1 and OUT2 should be roughly 90° out of phase; it should make a circle or something close to it on the screen of your oscilloscope when set in XY mode. It is controlled by R9, R10, and R12. These voltages can be tied into "basic motor neurons" to create a smooth sine motion on the output shaft of a servo.



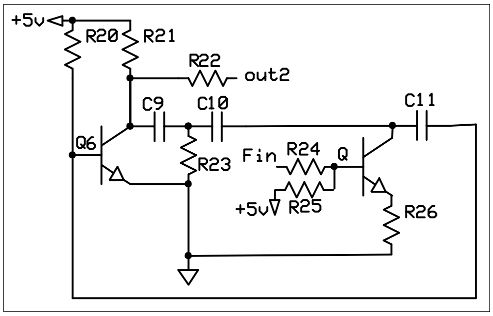
Circuit 2. Master/slave central pattern generator.

Having two servos phased locked roughly in quadrature forms the basis for locomotion. This can be a two servo walker, a single two axis leg controller for a quadruped, undulating motion for a fish or crawler, flapping, etc. (It should be pointed out that there are all sorts of circuits that can create sine waves or something close to it. A 567 tone decoder, for example, can be set up as a quadrature square wave oscillator and then a simple RC low pass filter is used at the outputs.) Computationally, the oscillators are acting as a sine (or other waveform) look up table.

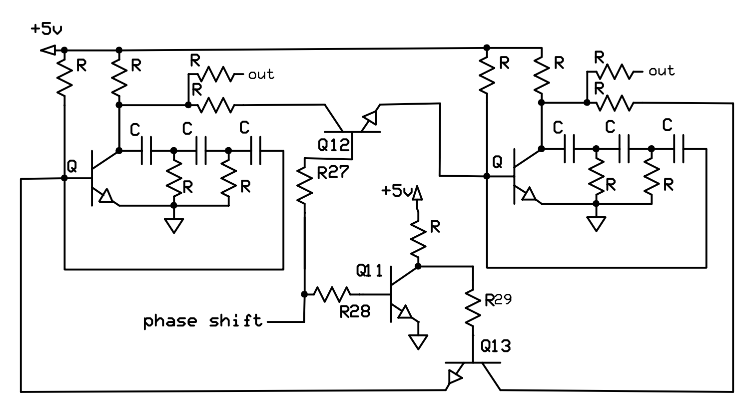
A word about the sine outputs and R9, R10, and R12. The values can be adjusted for quite complex wave forms. I have personally sampled and printed out periods 1, 2, 3, 4, 8, 16, and chaotic phase orbits. One setup showed a definite trend every 32 cycles, but never actually repeated.

This might be useful, for example, to keep a two servo walker from digging itself into or digging out of a hole on a soft surface (i.e., sand) by having a variable phase trajectory on the output shaft of the servos (of course, it might just make a bigger hole!). On top of that, the circuit can be adjusted to clip the ground or positive voltage, which might create some more useful waveforms. An interesting thing about weakly coupled oscillators operating in a chaotic mode is that they become information generators instead of just look up tables.

Circuit 3. Frequency modulated central pattern generator.



If one of the resistors in the RC network in the sine oscillator is replaced with a transistor that has a high base impedance to act as a voltage variable resistor, then the sine oscillator can be frequency modulated, as shown in Circuit 3. This would, for example, allow the robot to both walk and run by changing the oscillator time constant. By phase locking two of these circuits together, as in Circuit 2, more complex and interesting wave forms can be obtained by frequency



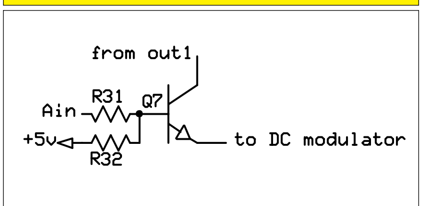
Circuit 4. Variable master/slave central pattern generator.

lampreys — one might try opamp phase shifters (say 10 or 20°) to allow for smoother undulation.

Circuit 5 shows a simple amplitude modulator. This allows one to control the amount of swing on the shaft of the servo. Its input is OUT of the sine oscillator and outputs to the DC modulator or directly to the input of the basic motor neuron. This is useful in quadrupeds to allow them to turn by adjusting the amount of swing in the front two legs. With two light dependent resistors that are set up as voltage dividers and a few resistors, a quadruped can have a light following behavior simply by adjusting the front leg swing proportional to light intensity.

Circuit 6 is a DC offset modulator that — when connected to the output of the sine oscillator or its amplitude modulator — allows adjustment to the part of the phase orbit that it oscillates in. For example, say your sine oscillator is set up for 2 volts, peek to peek. With the DC offset modulator, you can modulate it for a swing at 0-2 volts or 3-5 volts (minus NPN voltage drop), which affects where the servo shaft is swinging. (Is the control horn on the shaft swinging on the right side or left side?)

Circuit 5. Amplitude modulator.



This setup allows for positive input voltages for both higher or lower offset, which simplifies circuit design. This circuit is useful for tuning in two servo walkers, swimmers, and crawlers and can be used to adjust the balance point in a more complicated robot.

Circuit 7 shows a neat hack where you can replace the AM modulator and the DC offset modulator with the NE571 audio compandor — and get two channels, to boot! The output of the sine oscillator feeds into pin 3 and a voltage on pin 1 controls the amplitude (not pin 2, since this has a diode). Pins 6 and 7 are tied together and are the output that feeds into the basic motor neuron. Pin 5 with the resistor network controls the DC offset. The other side of the chip is hooked up in an identical fashion, but with a different sine oscillator feeding a different basic motor neuron. Use the resistor network shown as a starting point for inputs to the amp and DC modulators; add sensors later.

Pedal to the Metal

Trace 1 shows the chaotic leg gait of a light following robot that uses 34 NPN transistors and eight servos. The X and Y motions of the leg were plotted on a storage scope to show the up/down, left/right motion, as wells as demonstrate the drift in amplitude over time.

The actual circuit, additional trace captures, and videos showing this robot walking, are available on the *SERVO* website (**www.servomagazine.com**) for download. They should give you a good idea of the power contained in this approach to robot control.

Where to Go From Here

At some point, you might want to use microcontrollers for higher func-

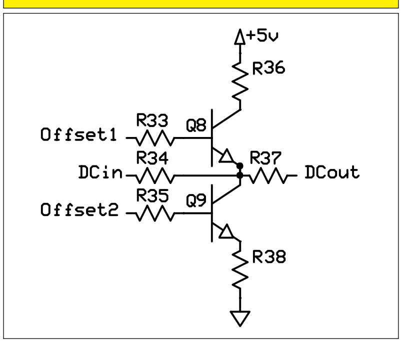
Suggested Reading

The Computation Brain
Churchland, Sejnowski

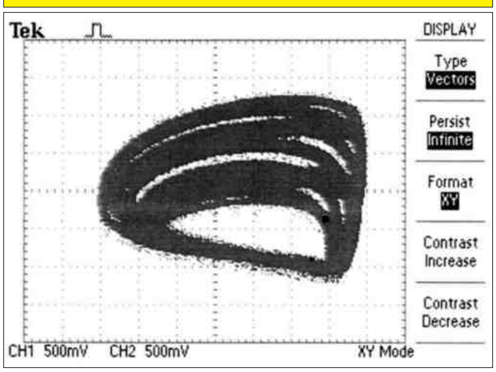
Chaos Theory Tamed
Williams

Self-Organization of Locomotory Controllers in Robots and Animals
(Doctoral Dissertation), Lewis

Circuit 6. DC offset modulator.



Trace 1. Chaotic leg motion over time.

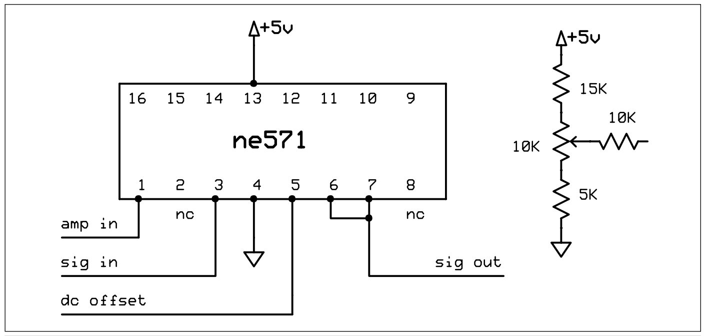


tions and think of the nervous system as one big, black box. This is actually a realistic approach, as we can walk and chew gum while maintaining balance without conscious thought. The brain doesn't have to deal directly with some of these lower functions, as can also be shown by the fact that a chicken with its head cut off can still run around; the central pattern generators for locomotion are actually located in the spine and some reflexes are actually monosynaptic (knee jerk reflex).

Sensors interfaced to the control system can provide a wide variety of adaptive behaviors, such as following light, avoiding an obstacle, or shifting a balance point. Overlapping of behaviors (subsumption) can allow for a wide range of actions with minimal circuitry. Some sensors — such as light dependent resistors — allow for a form of passive computing.

Once you have the circuits assembled to amplitude, frequency, phase, and DC offset, modulate the signal that feeds the "basic motor neuron." You can think of this as a neural cluster or a "synthetic ganglion" (ganglion being a term from neural biology). When properly set up, it becomes an analog computer that uses cheap hobby servos and offers a large amount of processing power per transistor.

These are the basics of this nervous net idea. It should be said that there is a patent pending on this concept. I've done a lot more research that can't be published at the time of this writing due to patent reasons. Research is being done on self-organization, short and long term memory, hardware genetics, and interfacing synthetic neurons to the nervous system — all using commercial off-the-shelf-parts. **SV**

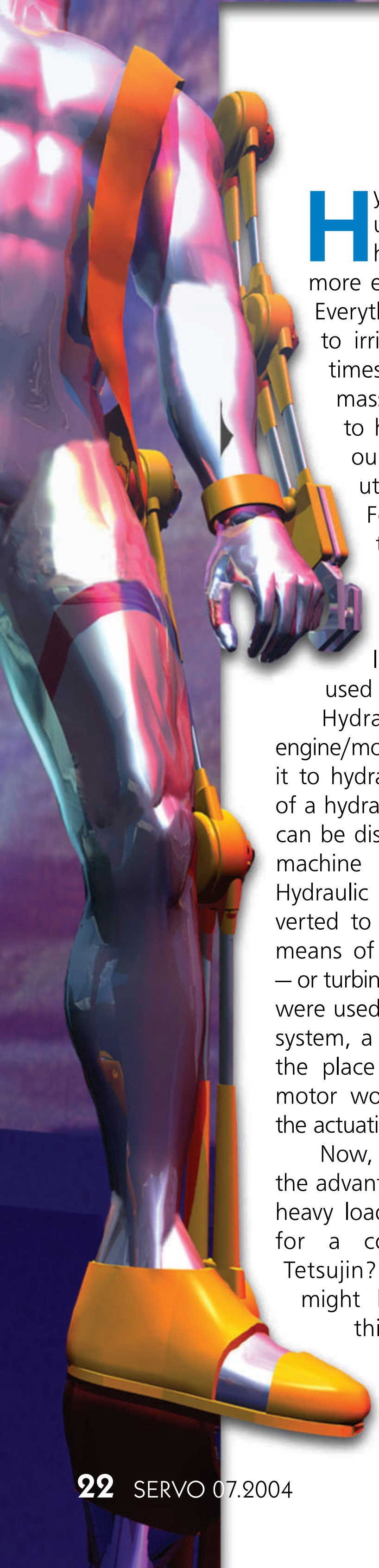


Circuit 7. AM and DC offset modulator.

About the Author

Tom Jenner is self-taught in robotics and electronics. His interests include artificial life as it applies to robots, neural computational theory, passive radar, and technical surveillance counter measures. He can be reached at robogeekboy@yahoo.com

TETSUJIN TECH



HYDRAULIC POWER PRIMER

by Bob Pitzer

Hydraulics have been used for millennia to help man move things more easily within his world. Everything from lifting water to irrigate fields in ancient times to water boiling in massive steam generators to help electrically power our modern world has utilized hydraulic forces. For this introduction, though, we'll define hydraulic power as a system where an incompressible liquid under pressure is used to transmit energy.

Hydraulic systems take engine/motor power and convert it to hydraulic power by means of a hydraulic pump. This power can be distributed throughout a machine by means of tubing. Hydraulic power may be reconverted to mechanical power by means of an actuating cylinder — or turbine. If an electrical system were used instead of a hydraulic system, a generator would take the place of the pump and a motor would take the place of the actuating cylinder.

Now, you may ask, what is the advantage of trying to move heavy loads this way, especially for a competition such as Tetsujin? Well, the answer might be simpler than you think. Basically, when you look at a simple force equation, $F = P * A$, where F = force, P = pressure, and A = area.

High pressure can build up in relatively small volumes (i.e., relative to area in this equation, which may coincide to a small displacement pump) with relatively low power input. However, by reversing the factors, the output force can be much greater because of the ability to increase the area in the output equation (say, by using a large bore cylinder) so, pressure would remain fairly constant. This way, a small power source in your exoskeleton could cause it to move massive external loads many times the weight of the actual exoskeleton.

Here are some other advantages of hydraulic systems over other types for this application (also applicable to aircraft type systems):

1. It is lighter in weight than alternative existing systems.
2. It is deadbeat — that is, there is an absence of sloppiness in its response to the demands placed on the system.
3. It is reliable; either it works or it doesn't.
4. It can be easily maintained.
5. It is not a shock hazard; it is not much of a fire hazard, either.
6. It can develop practically unlimited force or torque.

Many common hydraulic systems in the real world are huge, bulky, and capable of moving bridges. So, where do you look to find hydraulic components that might correspond to a low weight, strap-on-your-body exoskeleton that you could use for the Tetsujin competition?

There are many systems in the real world that utilize hydraulics for relatively light duty applications. Hydraulic lift gates on trucks are one application that has a multitude of suppliers; the components from these systems have been modified to fit a variety of other applications, including the hydraulically-operated "low rider" show cars you might see performing extraordinary suspension feats.

Many commercial lawn mowers have hydraulic controls for accessories that they use and — in colder climates — you may find hydraulics on snowplows that mount on the front of pickup trucks. Raiding junk and scrap yards for these types of components would be the low budget way to go; realistically, to save time, hydraulics are so ubiquitous that ordering the components you really want from online surplus or store front hydraulic supply centers won't cost much more (see sidebar).

Closed loop hydraulic systems can be designed in many different configurations, but typical systems have high and low pressure sides. The high pressure side is usually associated with the discharge of the pump; it may contain the directional control valves, accumulators,

shock suppressors, and pressure control valves. The low pressure side usually contains the reservoir tank that the pump takes suction from, heat exchangers, and oil filters. Some hydraulic components can be exchanged between the two sides. Study the diagrams shown to see how different components of a hydraulic system may be utilized.

There are many components that can be used in conjunction with hydraulics, so the following is a list of the basic components and their general uses with hydraulic systems. I have included my thoughts — gathered from years of working with hydraulics — on their effectiveness for the Tetsujin:

Accumulators: These devices do what their name implies — accumulate energy. They do this by working on the principle of gas expansion. Ideally, closed loop hydraulic systems work on the principle that the liquid used is incompressible; so, the volume you push down a tube via the pump is the volume that will emerge at the other end.

However, what if the pump you have isn't fast enough to get the fluid flow you want to operate an actuator at a fast rate? Well, in the actuator idle time, you could still use the pump to store energy to help push the actuator faster when you want to use it. Accumulators do this by having a tank that is half-filled with the system fluid and half-filled with a compressible gas, like nitrogen. The idea is that the tank system is at the system pressure of, say, a typical 3,000 lbs, but — when an actuator is activated and the pump can't make the volumetric flow rate for the desired actuator speed — the nitrogen will expand below the system pressure and still supply fluid to the actuators, at least until the accumulator has reached its capacity.

This may be mandatory for Tetsujin, since the application is to make a portable unit that is fast. To be portable, the power unit (pump and motor to drive it) may have to be small to get fast actuation; energy may have to be stored.

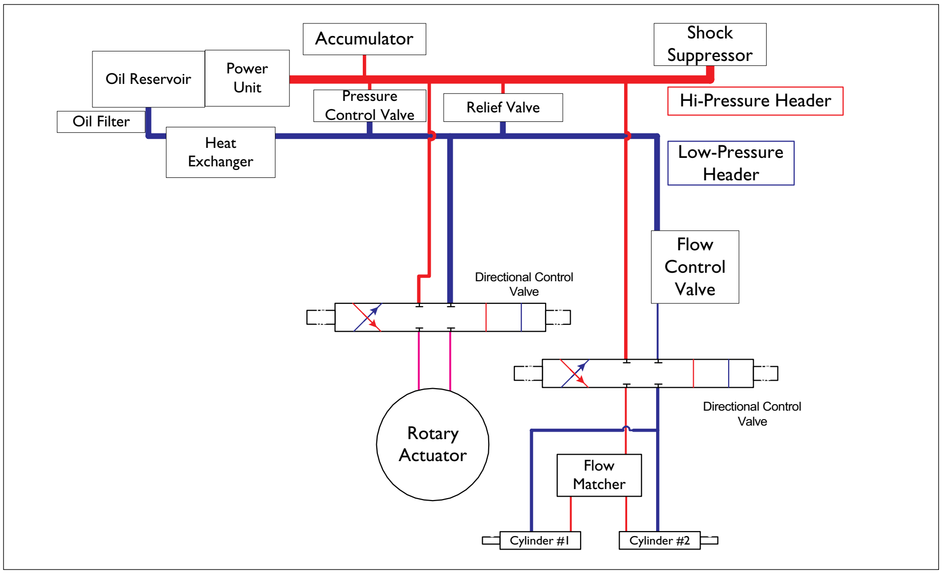
Shock Suppressors: These are basically accumulators, but their primary purpose is to absorb shock loads that might be introduced to the system. Being composed of incompressible fluid, shock loads on actuators may cause over-pressurization of hoses or other components. Providing a compressible element in the system will prevent this. Depending on the design of the overall system, this may be useful for the competition, but it's hard to tell without reviewing the design.

Heat Exchangers: Hydraulic heat exchangers are used in continuous-use systems, where the fluid is being constantly circulated, causing system heat build-up. As Tetsujin machines are not continuous-use devices, it's likely that the use of heat exchangers won't be necessary.

Hydraulic Cylinders: These are typically used to convert hydraulic system pressure into linear motion. Depending on the design of the appendages used for the lifting device, these may be mandatory for a Tetsujin machine. Typically, by using mechanical advantage, cylinders can produce more force than rotary actuators due to the available area for the fluid pressure to apply force upon.

Directional Control Valves: These are used to regulate flow to actuators or reverse the flow from one side of an actuator to the other for reversing the direction of a rotary actuator or expanding and contracting a hydraulic cylinder.

Flow Control Valves: For controlling the rate at which actuators move, you will use a flow control valve. Usually, they restrict the flow rate of fluid at one of the ends of an actuator, allowing friction to slow the fluid flow. This may be useful for controlling the speed at which appendages move or matching the velocity of actuators to move at the same rate.



Power Units: Typically, this refers to a package that includes a prime mover — such as a motor or an engine — combined with a hydraulic pump in one unit. This would be characteristic of the units used for the "low rider" car hydraulics. These compact, off-the-shelf type units and are ideally suited for Tetsujin. Some models have options for installed control and relief valves right on the power unit.

Pressure Control Valves: Sometimes referred to as recirculation valves, these are installed in systems with continuously running pumps — such as high capacity systems. They are normally regulated by spring pressure or are electrically actuated by a control system. The application likely doesn't apply to Tetsujin because it lends itself to smaller, on-demand systems controlled by a pressure sensor or control valve-activated motor control.

Pumps: Pumps are apparatuses for building pressure in the system, usually actuated by a motor or engine. There are alternate ways to pressurize a closed loop hydraulic system — for example, with gas — but the most likely system used in conjunction with Tetsujin will have a pump. The pumps are available in many different pressure ranges and capacities and selection will depend on your system.

One of the better ways to figure out what might be required by your system would be to consult a local hydraulics supply house. Parker Hydraulics has representatives in most of the larger metropolitan areas around the US, but, due to the wide-spread usage of hydraulics, there should be some sort of local supply house that can assist in specifying what you might need to make your system successful and optimized.

Oil Filters: These help keep the fluid clean to avoid unnecessary wear to system components from foreign objects in the oil. These are typically installed in systems used in dirty environments or ones that are in continuous use. Being that a Tetsujin machine will most likely be a low cycle, momentary use apparatus, oil filtration will not be necessary, but serious attempts to keep foreign objects out of your closed loop system during assembly should be made to make sure you won't have objects in the system that may block flow passages or cause a valve malfunction.

Pressure Gauges: These gauges visually identify what the fluid pressure of the system is; this might be useful for identifying problems. They are not really necessary, though. They can be placed on the outlet of pumps to identify maximum pressure or at actuators to see how much pressure is actually being delivered at the desired flow.

Most power units have built-in relief valves that lift at a set pressure, making sure the system pressure never exceeds what may be allowable by system components. Most "lift gate" power units typically are set to run at ~ 3,000 PSI, but this is easily changed by installing a new spring to control the lift pressure.

Rotary Actuators: Rotary actuators are meant to actuate in a circular motion around an output shaft. This might be useful for something like an elbow joint on a robot arm. They are useful for tight packaging applications where space might be at a premium. While hydraulic rotary actuators can make a lot of torque, they are more expensive and can't produce as much torque as possible with linear hydraulic cylinders using mechanical advantage.

Implementation may be more difficult because the ability to have a secure attachment on the output shaft requires more fabrication to provide a safe joint. Also, the rated cantilever loading on the output shaft of the actuator may require additional fabrication of rotary joints to ensure that the mechanical limits of the output shaft aren't reached; this could either break the shaft or cause shaft bearing failure.

Applicable usage for Tetsujin will be dependent on the design of the machine, but special attention needs to be paid to making sure actuator output shaft mechanical limits are not exceeded.

Motors: The most recognizable use of hydraulic motors might be those used to drive the tank tracks on vehi-

HYDRAULIC RESOURCES

Wholesale Hydraulics
http://wholesalehydraulics.com

Surplus Center
www.surpluscenter.com/hydraulic.asp

Northern Tool
www.northerntool.com

Gauge Magazine
http://store.gaugemagazine.com/index.asp?PageAction=VIEWCATS&Category=263

Northern Hydraulics
http://northernhydraulics.sofastweb.net/ws/aboutus3.php?page_id=5374&OVRAW=Vickers%20hydraulics&OVKEY=vickers%20hydraulics&OVMTC=standard

North American Hydraulics
www.nahi.com/

Bosch Hydraulics
www.boschrexroth.com/country_units/america/united_states/en/products/bri/index.jsp

Greer Hydraulics
www.parker.com/ead/cm1.asp?cmid=101

Nachi Hydraulics
www.nachi.com/hydraulics/

Parker Hydraulics
www.parker.com/hydraulicsgroup/indexv4.asp

Denison Hydraulics
www.denisonhydraulics.com/

cles — for example, bulldozers. They are meant to run in a continuous rotary motion. Tetsujin not offer applications, since it is dedicated to linear translation of the lifting weight, i.e., lifting the weight up into the air. Creative uses may be possible.

Flow Matchers: These are known by a couple of different names. They are meant to match two or more actuators in synchronous, parallel motion. This is useful for getting two identical cylinders to run identically; they could be used for having two different appendages lift in synchronous motion, although having actuators locked together like this will not allow independent motion of the actuators.

Relief Valves: Typically, relief valves are used to protect system components from overpressurization, which is a very important feature of any closed loop system. Since the fluid in the system is ideally incompressible, if pressure continues to build for some reason — for example, a pump being stuck on — the weakest component of the system will eventually fail.

The intent of the relief valve is to be the weakest component and to relieve the pressure before it breaks a more integral or valuable system component. Most power units have an internal relief valve built in, but — when assembling a system — the engineer should ensure that there is a relief valve somewhere in the high pressure side of the system.

Oil Reservoir: This is the reserve volume for the system. Usually kept at atmospheric pressure, it's associated with the low pressure side of the system. Fluid returning from actuators drains into the tank and waits to be recycled by the pump for reuse in the system. It makes up for the leakage associated with hydraulics, too. Don't fool yourself — hydraulics always leak.

I have designed and maintained hydraulic systems for the better part of 17 years now and have never seen one that didn't have puddles of fluid around it. Just have lots of rags handy to clean up the mess. Also, be aware of the fact that, if you have the pump taking a suction off the reservoir tank, the pump suction pickup will always remain submerged, regardless of the orientation of the oil volume.

These are just a few examples and ideas of what you can do with hydraulics for a Tetsujin entry. The application principles are nearly as limitless as the ideas that are being formulated for moving the weight around. There are endless online resources for researching what you want to do with your system. Search engines are a great start.

Also, as mentioned before, if you are really interested in applying hydraulics to your entry, visit your local hydraulics supply center and tell them what you're up to. Who knows, maybe they'll be willing to supply you with discounted or free components just because they think the idea is cool! **SV**

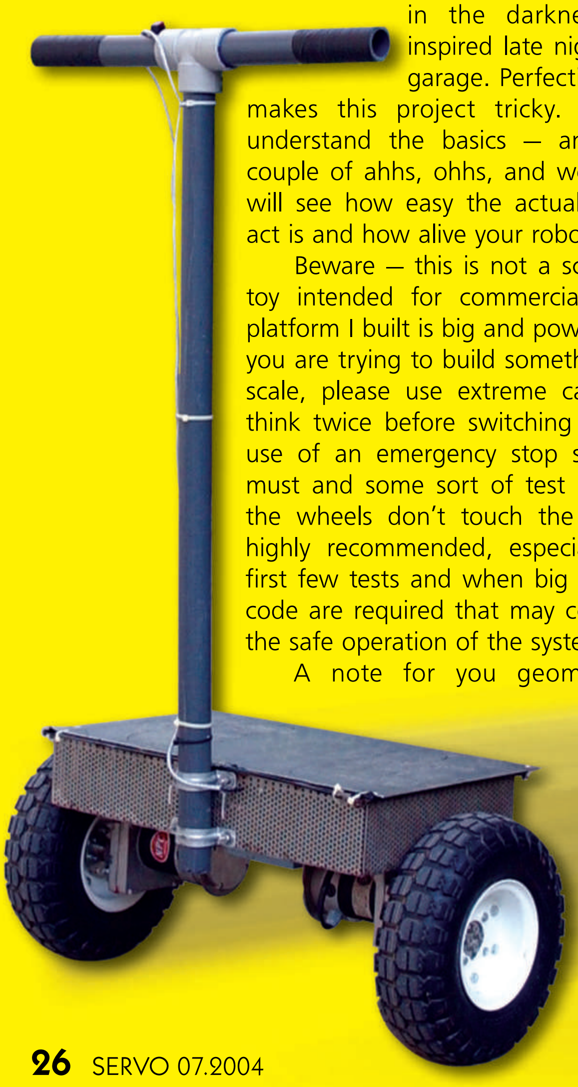
ABOUT THE AUTHOR

Bob Pitzer is a mechanical engineer who graduated from the University of Florida in 1997 after six years in the Navy as a Nuclear Certified Machinist Mate on a 688 class submarine. For five years, he worked with Intel, designing and building semiconductor manufacturing equipment before undertaking independent contract design. He is the co-builder of combat robots (**www.raptorrobotics.com**) and the owner and producer of the BotBash (**www.botbash.com**).

Two Wheel Dynamic Balancing

So, what does it take to balance a robot on two wheels? First, let me start by saying *patience* and, in my case, *time*. Second, let me add *attention to tiny details*. If you want a quick way, you could just add a third wheel and life would be simple again, but, I guess that if you are reading this, this is not your desire.

It is not a difficult thing to do; it is, however, challenging and tricky, especially if you want to do it reliably and safely. It will take a lot of trial and error, but, once you are up and running, you will feel great about having made such an accomplishment and it will keep you working toward perfection. Also, it is a jaw dropper for friends and family. Your loved ones will at least know you are doing something cool while in the darkness of an inspired late night in your garage. Perfection is what makes this project tricky. Once you understand the basics — and after a couple of ahhs, ohhs, and wows — you will see how easy the actual balancing act is and how alive your robot will look.



Beware — this is not a scooter or a toy intended for commercial use; the platform I built is big and powerful, so, if you are trying to build something of this scale, please use extreme caution and think twice before switching it on. The use of an emergency stop switch is a must and some sort of test bed where the wheels don't touch the ground is highly recommended, especially in the first few tests and when big changes in code are required that may compromise the safe operation of the system.

A note for you geometry lovers: "zero angle" does not means "zero degrees," it means the rest angle or the angle at which the platform is stable (desired angle) and the wheels are located right under the center of gravity and everything is in balance. It's important to remember that this is what zero means in this article.

Hands On

Platform balance is achieved in the same way you balance a broom or a stick in your palm, only, in this case, the problem is constrained in one single axis. As a result, if the balancing object tilts forward, your hand needs to move forward to kick the object's tip back; if it goes the other way, you do the same thing in the opposite direction. The difficult part of balancing a broom or a stick is over- or under-correcting; the same principle applies to balancing a robot, only the brain is an MCU or some sort of computer and the sensors are not visual (your eyes), but an IMU (Inertial Measuring Unit), which consists of inclinometers and other sensors that read a dynamic behavior of the system.

Think of a balancing robot as one that keeps its wheels under its mass weight or center of gravity. You push it one way and the wheels will have to move in to that same direction to keep its weight supported over the mass center. To make the robot move forward, you just set an offset to the zero angle that is a bit off to the desired direction and, since it has to keep the platform on that angle to prevent tipping it, the robot has to keep moving forward to sustain the desired angle, not allowing the platform to tilt more.

If you were trying to balance a broom on your hand at, say, 10° of center, you would have to constantly move forward to keep it at that angle. Since the broom does not have a counterweight to keep it from tilting, your

Robot

forward movement is actually acting as a counterweight (or counter action), which is actually translated from inertia. If you keep just the right speed (in theory), the desired angle will be sustained. If you move faster, then the angle will be overcompensated or overshot. If you move slower, then it will be impossible to maintain that angle and gravity will take over very quickly, so the speed has to be just right to sustain the desired angle over time.

To make a long story short, the angle offset will be proportional to the desired speed. Faster speeds will mean that the platform will have to be at a stiffer angle. For slower speeds, it will have to be just a bit off from zero angle and, to stop it, it has to be right at zero or right on its center of gravity. This same principle applies in the other direction (going in reverse). The limits are also proportional to your motor's abilities to have enough power to sustain a desired angle or to compensate for a big change; in addition, the platform design plays a big role in how fast it will tilt or respond to the applied motor torque (especially weight).

The batteries need to be strong enough to deliver the power and the motors need to be ready to compensate for big changes (or large payloads). If you want to drive your robot on even, smooth surfaces, then you will not need to keep a lot of overhead to compensate for bumps or other un-even properties. If you want an all-terrain robot, you will have to have a lot of overhead in the batteries and motor power, as well as in the PWM driver (headroom).

Another important factor is the sampling rate of your sensors. Your bot will need to know as fast as possible what its current angle is in order to compensate. In my experience, more than 20 samples per second or 20 Hz will be okay, but a bit jerky. In my design, I sample at around 60 Hz and it works just fine. Others suggest sampling at around 100 Hz. The sampling rate is not only limited by the MCU's speed, but also by your sensor's bandwidth.

If your sensors are not fast enough, but your MCU is fast, it will read that the platform is at the same state as it previously was, even if it changed position. Usually, sampling rate problems are more often related to the MCU's sampling speed, rather than the sensor's bandwidth or sampling rate. This is the case because the filters used here not only take MCU cycles, but also need some time to get an accurate reading and a good, precise, and smooth interpolation, since filters are based on a model and the history (and a lot other things).

How to Determine the Robot's Present Inclination or Angle

There are a variety of sensors that can determine the current angle. Some of them are insanely expensive (the better ones) and some are within the reach of the hobbyist. Several sensors of different kinds with different fundamentals can provide the MCU with the current attitude information (*For those of you who aren't familiar with*

by Francisco Lobo

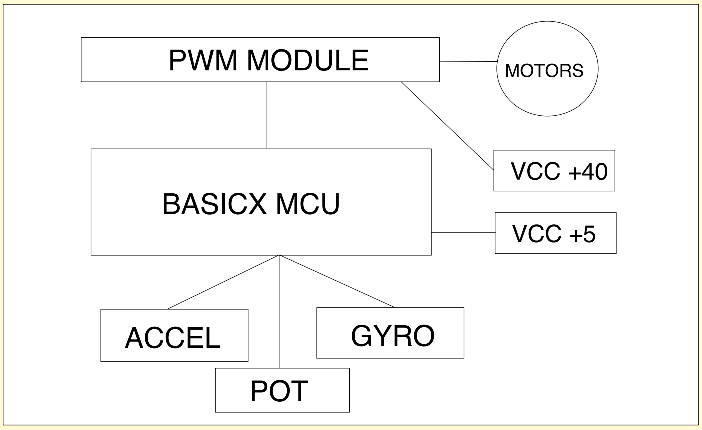


Figure 1. A block diagram of the control system.

aviation terminology, that means "tilt."). All of them have some sort of catch in their implementation. In some cases, the catch can translate to a lot of work for you, since you need to keep filtering the signals. In other cases, the catch can translate to a lot of money for a better sensor with greater quality. In both cases, some filtering needs to be done. I predict that, in a few years, sensors (or sensor systems) will be available with little or no drift for less than $100.00.

The sensors I used are most commonly seen in other systems, including camera stabilization, vehicle stabilization, and cruise control systems. They are not that expensive and are within the reach of any hobbyist or enthusiast. The rule is, the more money you put in, the better sensors you will get out — better meaning not only less drift in the case of gyros, but also less noise, a faster sampling rate, etc. Bear in mind, however, that this does not necessarily mean that you will have less work to do on the filtering.

There is no rule on what to use to determine the tilt angle. Some people use proximity sensors arranged to read the distance from two points on the platform to the floor. Others use inclinometers only, while still others use compound sensors to filter each other out. The truth is that there are many ways to do this and the best way will depend primarily on your application. I chose to use a dual axis accelerometer and a gyro.

Figure 2. The author demonstrates his prototype.

The accelerometer reports a vector to gravity and, thus, the angle of the unit with respect to the ground. (Don't get confused; this is not an error. With this type of accelerometer, it does report a vector to gravity.) This accelerometer works as an inclinometer and the signal is very clean and precise. The catch of this sensor is that, when moving forward (linear acceleration), they start spitting wild values. They are very sensitive to motion — and who can blame them, since they where born to be accelerometers, not inclinometers. They are great while the unit sits static, only tilting, and not moving around.

The accelerometers can be read in different forms, depending on your actual sensor. I chose the analog output, proportional to the actual tilt. Accelerometers are very precise when reporting valid readings in the right conditions. Keep in mind that, the closer you mount them to the pivot point, the more accurate they will be because the platform will not throw the sensor forward when tilting.

The gyros report the rate of turn and they tend to drift with time, since they don't know what the direction is to gravity and are affected by temperature and other ambient factors — including extent vibrations, to some degree. The gyros don't tell you what angle the robot platform is at, but rather how fast it is tilting toward an axis. Thus, by integrating its output, you can get the actual tilt angle. Oh, yes — with a drift. The gyros I used send their values to the MCU through an analog output with a voltage proportional to the rate of change.

To integrate the gyro, you can use the following formula (or something similar, since it is very simple). Be aware that the result will be the actual angle, but it will be biased and will drift over time. So, you need to remove the bias and correct for the drift to get the real angle value. Also, DT should be measured accurately, as this integral sum will soon report wrong angles as the errors accumulate.

The formula: $A = A + (G\_ADC * DT)$

where A is the actual angle, G_ADC is the ADC reading of the gyro, and DT is delta-T — the integral processing time.

DT can be set in two forms, either by determining the difference in time from the last sample to the actual one (this method is preferred) or by estimating it as: 1/samples per second (ex. 1/100 Hz = 0.01 seconds). If you estimate it, make sure that you get really close to reality, otherwise, your integral will not work well. If you have access to timer overflow interrupts in your microcontroller, then you can estimate it very closely and get better results.

In my case, I reduced the resolution of the reading to get a more steady integration, since the gyros also send some noise. So, I sacrificed a bit of precision (not necessary, but still precious, precision) and gathered a more stable reading.

Since the gyro integration will work okay with some drift, you have to de-drift the integration somehow. This is where filtering comes in handy. If you use a good filter, it will also remove the noise from your sensors. Let me tell you that

this is where the fun begins.

If you fuse both sensor readings together, you can get a "filtered" output with a very precise value for the actual tilt angle. Several filters are used in this type of application to "de-drift" and "de-bias" the gyros and co-sensors; they also optimize the fusing. One of the most common, well documented types is the Kalman filter, which uses a model of the sensors and a least squares function — among other calculations — to "guess" what the closest value to the true angle is.

A lot of material is written about this filter and it is used in a vast number of applications. A great deal of documentation exists online and many books are available about this filtering method. To be completely honest with you, at this point, I don't fully understand it. What I did in my robot as a temporary solution was to use the gyro integral for the current angle and the long term average of the accelerometer to de-drift the gyro. To the best of my knowledge, this idea was first proposed by Trevor Blackwell and it works very nicely, but right now I am working hard to implement a fully "optimized" Kalman filter to get the actual angle.

The filter I used is more than enough to make your robot balance on two wheels for a long, long time, but, like I said earlier, precision and smoothness is what matters in this project. Other filtering methods I have heard of are based on fuzzy logic and — most likely — neural networks; however, I don't really know how they are implemented, in practice. The filter implemented in my code is very stable, but it gets a bit jerky when used to hover the robot in a fixed place; sometimes this is minor and unnoticed.

I am not going to go into much detail on the physics behind the sensors, since space is limited and I want to provide as much information as possible.

The IMU sensors I used are coupled together in a board. I used one by Rotomotion; I only used their sensor interface boards, which are primarily made with one analog device (ADXL202). I also used a MEMs accelerometer and an NEC-Tokin (CG-16DO) ceramic rate gyroscope, both of which are filtered and amplified to provide a clean reading.

***Note**: The hardware filter provided with the Rotomotion board is not a Kalman filter. It is an analog, low pass filter and amplifier (opamp) designed to reduce noise from the sensors and scale the gains. You will still have to make your filter via software to get a good angle estimate.*

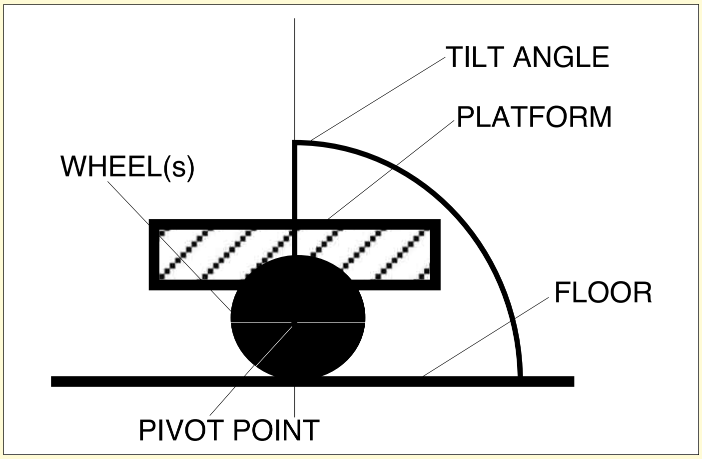


Figure 3. A physical model with definitions.

PWM Module, Motors, and Batteries

You will need to have enough power to compensate for quick changes in angle, so choose a PWM drive and a set of motors that will give you more than enough power to balance the robot. Keep in mind that less weight is better, so make a good balance with these factors. Whether you design your own module or get a specialized one for this, make sure that you will have sufficient power to work with.

Mine is made by Roboteq (see Parts List for details) and, in my case, it is commanded by an RS-232 port from my MCU, but an analog source is more likely to do a better, cleaner, and quicker job. The Roboteq module has the benefit of a wide variety of options for controlling it and it has a lot of spare power for the application. The typical application for the Roboteq module is for combat robots, but, if you are a hobbyist planning to build large bots, this is a great investment to have in your shop. If you are making a small robot, this driver might be too much. (If you are making a huge, ultra, mega big bot, this might not be enough!) The point is that you will need to choose the right type of PWM and motors to successfully counteract the real world physics applied to your robot.

The motors, wheels, and hubs in my project are provided by NPC Robotics and are made primarily for electric wheelchairs. (See Parts List for details.)

Batteries play a very important role. Your batteries are the heart of your project and they need to support whatever power is sourced by the PWM unit, so make your numbers work before you start implementing PWM, motors, and batteries.

The wheels are also important, but they are not critical,

PARTS LIST

1 —	Controller/Brains BasicX MCU BX-24 (**www.basicx.com**)
1 —	Inertial measurement Rotomotion Z "sensor only" board (2 DOF) (**www.rotomotion.com**)
20 —	Batteries 7.2 VDC 3,000 mAh NiMH from R/C cars (**www.batteryspace.com**)
2 —	Motors by NPC Robotics (**www.npcrobotics.com**)
1 —	AX2550 Dual PWM module by Roboteq (**www.roboteq.com**)

Note: The batteries were hooked up five in series and then in parallel with four more banks of serial batteries using a bridge rectifier between banks to prevent them from discharging each other when they don't have the same balance in charge.

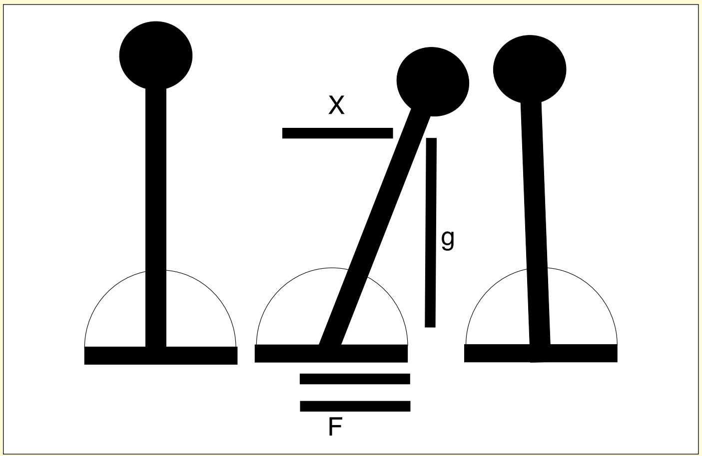


Figure 4. Forces acting on the balanced load.

so long as you use common sense. The larger the wheels, the more momentum you will have. The more momentum, the more traction, and the more traction, the faster you can respond. However, if you choose a very large wheel, your robot will balance even before moving forward without allowing it to tilt to navigate. So, it's a "balance" between the PWM module, motors, gears (backlash), and wheels that determines how smooth the robot's movement can be. If the wheels are too small, the robot will be a bit too jerky while in hover.

If you don't mind your robot being a bit jerky in hover, the gears can be standard. If you want superior precision, then your gears need to be low on backlash; however, for normal operations with very small and maybe unnoticed jerkiness, you can get away with standard gears. The ones I used came attached to the motors by the factory (see Parts List). Clever programming can work around lots of physical problems. Keep this in mind!

The Frame

The frame — or platform — I used in the prototype is less than optimal, but it works. It is a simple, stainless steel box, but this will soon change to a temporary, fiberglass chassis and then to its permanent, carbon fiber chassis. Both of these were provided free by some associates who do a lot of work with these kinds of materials. I recommend doing your tests with some kind of hard platform, like aluminum — or whatever might define "hard" for your robot's scale.

When choosing a material, bear in mind that it will hit the floor hard (no matter how good you are) and, if the material is not tough enough, it will break. I discovered is that, as long as you use strong common sense, the first balancing act will be forgiving. Just try the best you can to use precision from the beginning — along with good quality materials and precision in your algorithms — since this will result in greater performance and smoothness potential later.

Figure 5. Note that all the weight is *above* the wheel axis.

The Brains

For brains, I chose NetMedia's BasicX microcontroller; it gives me the ability for rapid development with a great deal of power. It runs wonderfully and makes the prototype easier, since it requires fewer parts than other MCUs (actually, almost no additional parts). The Basic language also provides good, readable code and it can be programmed from any serial port from your computer, straight out of the box.

The MCU is very well-made and powerful enough for this and other applications. It allows you to start solving the real problems right away instead of setting up a peripheral — you just plug and play. The BX-24 also has enough memory to support a much bigger program than the one required by my application, so there is room for expansion. Tech support is also great, so I decided to use it.

The Software

The software I wrote is actually not that large; the actual balance routine was placed in less than 50 lines of code. As I mentioned earlier, it is in Basic and I am using some custom serial port routines to output values to the PC console and to the PWM driver at the same time. The Kalman filter will bring the MCU to its knees, but I am predicting that — with optimization — it will be powerful enough.

Put It All Together

The hardware part that is used for control is very simple. It actually does not use any components that are not found onboard the BX-24; the only extra, low power components are the sensors and the pot(s) that make the unit turn.Since the robot doesn't have any intelligence or autonomy built in, I used a set of potentiometers to steer and drive it around.

I recommend using a separate battery to power the logic; also, make sure that all your grounds come from the same place (PWM module, MCU, batteries, etc.).

The connections between these modules are fairly simple. You read your sensors through the ADC pins in the BX-24 (pins 13-16) and talk to the PWM module with either a DAC port or through the serial port, as I did. The serial port of the Roboteq controller is configured at 9600 baud, one start bit, seven data bits, one stop bit, and even parity. So,

you will have to use the COM3 of the BX-24 in order to configure the serial port in this fashion.

To talk to the Roboteq, you are required to send 10 "CR" characters to slave it to serial port command mode and then you will have to split data to it — even if it's garbage, it needs data within two seconds to keep the watch dog protection from kicking in and shutting down the motors. You could turn off the watch dog protection, but I don't recommend it — should your MCU fail, the unit can go mad and do crazy stuff. If you keep the watch dog timer enabled, it will offer extra safety in the system. So, it is worth the difficulty of sending data constantly to the unit to keep this protection.

If you plan to use a lot of power, make sure to use the right cable and connectors. The emergency stop button — or dead man switch — is connected to the supply lines of the Roboteq unit. This unit has the option to supply its voltage either from an external source or from the motor batteries. I use the motor batteries and, in this configuration, the two external supply wires can be used as an off switch if you short them. I connected two switches in parallel for the kill switch — one for normal ON/OFF operations and the other one for the emergency stop.

The ON/OFF is very useful when testing the MCU, as you do not want power in your motors. I used another battery (13 volt) to supply the voltage to the MCU (the BX-24 has an onboard regulator for +5V). The grounds are joined together from the motor batteries, PWM module, MCU, and the chassis. This will ensure a proper common ground. If you don't have proper grounding when testing, be careful when you unplug the serial port cable or external power supply; it can make your robot do crazy things because the ADC is very dependent on the reference to zero and this reference comes from the ground. Also, if you don't use a BX-24, be sure to check your settings for the Aref (analog reference) pin.

The IMU sensors are mounted very close to the pivot point of the unit. This helps prevent the accelerometers from moving forward when the unit tilts. If you put them on the top and the robot tilts, it will suffer from some linear acceleration due to the arc that is produced. The closer the IMU is to the pivot point, the fewer problems it will have with linear accelerations.

When the IMU sensors report an angle, the controller needs to immediately apply a proportional force in the same direction in order to restore the platform's balance, as seen in Figure 4, where:

F = The proportional force provided in reference to the angle.
G = Gravity.
X = The resulting phenomena caused by moving the motors toward the angle error.

Also, notice that when the motors move in a certain direction, they produce counter torque and this force is applied inversely proportional to the direction the wheels are turning; this force will move the mounting plate (in this case, the platform).

How fast the wheels move is also determined by their momentum. When torque exceeds the wheels' momentum, they rotate in the opposite direction; the more torque required to compensate the angle, the faster the wheels will move in a certain direction to create the needed torque to keep the platform from tipping. So, in reality, the balancing act can be explained in many ways and this will depend on how you visualize and attack the problem, but the resulting solution will most likely be the same. A force is required to counteract the angle error and that force needs to be inversely proportional to the force that is tilting the platform.

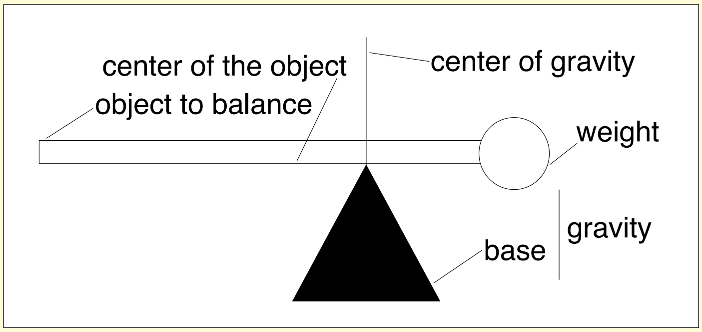


Figure 5. Balancing the load.

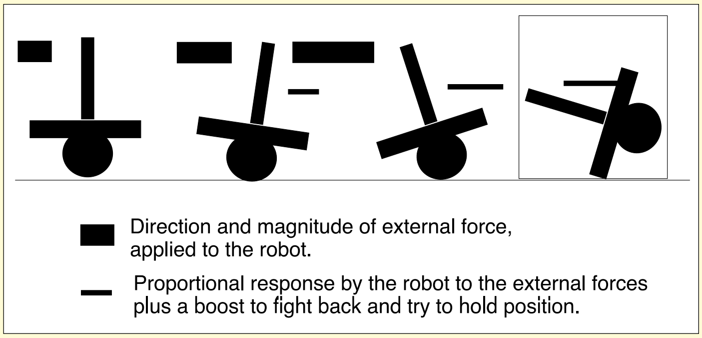
One of the forces that can change the angle of the platform is gravity — someone pushing it or a change in the center of gravity. That's why it's called a dynamic balancing platform — it constantly compensates for changes in the angle to keep the unit in balance. This might mean that the platform itself may not be leveled or parallel to the ground, but, when the motors produce zero torque and the angles no longer change, then the unit is in balance.

If the desired angle is not in the center of gravity, then the motors will need to move, tilting to keep the angle in the platform as close to the desired angle as possible. If you want the platform to hover, the variable "center of gravity" in software will be equal to the physical "center of gravity" of the unit; however, if you want it to move forward, the software variable will be a bit off from the physical one. How little or how much off it is will determine how fast the robot will go.

If this sounds complicated, don't get scared; it's really more simple than it seems. Let's illustrate another example to demonstrate that "level" doesn't always mean "balanced":

Notice how the stick is not exactly at its measurable center, but a bit off to the left to compensate for the weight that it added on the right cap of the stick. This same principle

Figure 6. The result of overreacting to the reaction force.



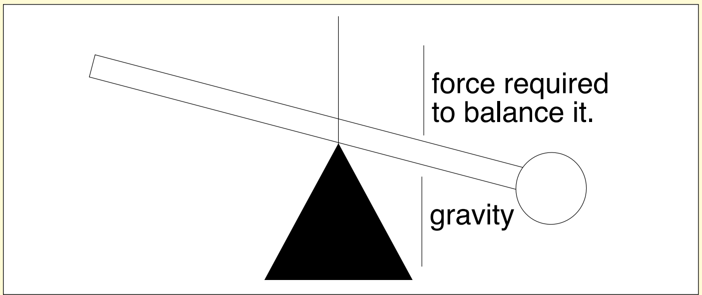


Figure 7. Restoring balance by applying a force.

applies to your robot; the true center of gravity might not be level or parallel to the floor, but a bit or a lot off; this has to be either calibrated or coded to the robot in a fixed or dynamic way where the robot will learn where its CG (center of gravity) is by measuring the torque required to keep it from tilting. When the torque is zero, then the unit is in balance.

The more practical way is to find this measure for yourself and hard code it to the robot. This way, when the unit is booted, the robot will slowly try to catch the angle that will balance it. When you want it to move forward, then set the offset of the real center of gravity proportionally off to the speed desired. If the robot is not in its real center of gravity, the motors will try to compensate and this is what makes the robot drive forward.

There are some other factors involved. You can't just set the torque proportional to the angle and expect it to work. It has to be damped somehow to avoid putting the unit into resonance or not having enough torque to compensate. This is done by means of "rate of change," which is provided in a great form by the gyros. If we push the stick faster, then it will need to react with more force than that applied in order to compensate. Remember that with an action comes a reaction and, in order to keep it in balance, the force exerted has to be proportionally inverse to the one applied. This will set the body in rest.

In other words, the variable that sets the torque to the motors is based on a PID (Proportional Integral Derivative). The PID provides a proportional value — an integral that allows it to amplify that value and a derivative that allows you to damp the response based on some history. PID is very useful and can be complex and simple at the same time.

RESOURCES

The journal, videos, and all resources — including source code and schematics — can be found on my site:
www.fusionglobal.net

The robot I built uses a lot of the parts and algorithms found on the scooter by:
Trevor Blackwell — **http://tlb.org/scooter.html**

Much of the research material I used on the subject can be found at:
www.tedlarson.com/robots/balancingbot.htm

http://geology.heroy.smu.edu/~dpa-www/robo/nbot/

http://autopilot.sourceforge.net/

If you would like to learn more about the Kalman filter I am implementing, go here:
www.cs.unc.edu/~welch/kalman/

If you want to contribute to the project, please Email me:
francisco@fusionglobal.net

Implementation

Algorithms are usually calibrated in different ways. The types of constants we are going to use are usually for the amplification of the gains. There is a way to actually calibrate the constants without coding back and forth. Remember the steering pot? This pot can serve well as a gain calibrator.

After you are comfortable with the gains, you can hard code them in your source code or leave the pot for further gain calibrations. You could also implement a fuzzy logic or something that will auto calibrate your platform, but this will be more complex than what is required for this app.

The algorithms I used for torque are based on the following, simplified pseudo-code:

```
Torque = ((CurrentAngle * K1) * (Rate * K2)) * Gain
```

where CurrentAngle is the actual filtered angle and Rate is the rate of change. Gain will be the feedback loop (usually based on history to dampen or amplify the result) and K1, K2 represent the normalization constants for each sensor input.

There is more than one way of balancing a platform. Some of them involve more complex mathematics. What I found is that, when you are trying to balance a platform for the first time, you don't need complex mathematics; you just output to the motors a value based on an actual angle times the rate and you will have enough torque to counteract the physics applied to your robot and, thus, keep it in balance to some extent. From here, you can start working on making it stiffer and more stable.

Now we'll take things a bit further. In my experience, making K1 change in proportion to the current angle makes the platform a bit more stiff and resistant to change, but be careful — too much and it can set the whole unit in resonance. So, here is what I did:

```
GainAmp = Abs(CurrentAngle) * K3
Torque = [(CurrentAngle * (K1 + GainAmp)) *
            (Rate * K2)] * Gain
```

Be careful to make sure that GainAmp does not reach zero; you will have a multiplication by zero, causing CurrentAngle to be zero, which is normally not true. In my case, I prevented GainAmp from becoming less than 1.0 with an IF statement:

```
If GainAmp < 1.0 then
       GainAmp = 1.0
End if
```

Also, be careful with your variable types and conversions; some of them truncate your values and reduce resolution or produce unwanted results. I usually sample ADC in integer values (0 to 1,023) and then convert them to floats or singles to make it easier to code and to make things more compatible with other trigonometric functions and equations. Make sure that your variables are scaled to match mathematics from other variables involved.

These are the basics of balancing; however, there are many other details that need to be covered in order to achieve a proper balancing robot that can hover on two wheels and also move around gracefully for days or months.

GainAmp is very useful to make the robot stiffer and tougher; however, you must be careful not to make it too tough or it will go into resonance and wiggle until it gets out of control. This type of attenuation from the feedback gain is not achieved through "GainAmp," but rather through the speed limit for a particular maneuver you want the robot to perform. This sets a tolerance and allows the "brain" to react accordingly and track if something goes faster than expected.

I track speed through an integral of the torque and so the tolerance is set by a speed limiter. The speed limit is dependent on what I am trying to do. If I want the unit to hover, then my speed limit is close to zero. If I want to allow the unit to move around, then my speed limit is set higher. This allows the unit to compensate for external physical changes. Speed can best be tracked through the use of encoders, but the integral works very nicely for this purpose.

With a robot, speed limiting is not that important, but, if we take the speed limit as a factor to compensate for external forces acting on the unit, then the robot most know how fast it is going and how fast it is allowed to go. If the unit goes faster than the tolerance, then it has an external force influencing it. If we want the robot to cruise at 5 mph, then the speed limit should allow 5 mph ± a threshold, but nothing more.

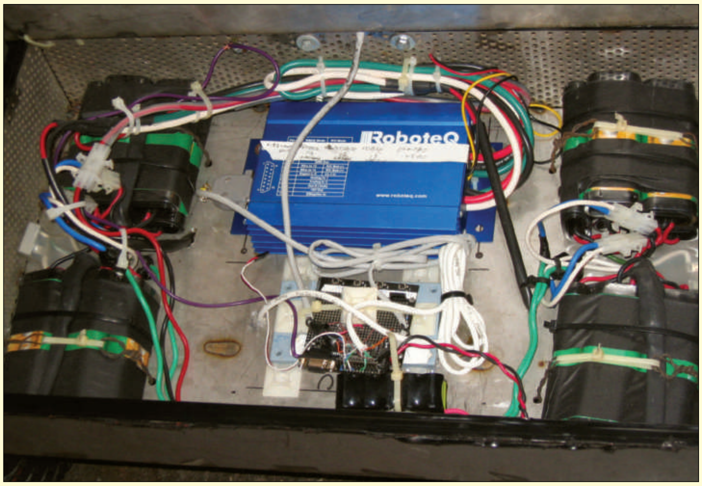


Figure 8. A peek inside: batteries, BX-24, and the Roboteq.

If we exceed 5 mph and the known angle to achieve 5 mph, then there is an external force affecting the unit — either a change in weight, someone pushing it, a bump, or something else. In order to limit the speed that the platform is allowed, we need to move faster toward the same direction, thus forcing the robot to tilt to the opposite side and then slow down the torque. Yes, you have to move faster toward the same direction in order to tilt the platform to the opposite side and then allow it to slow down. If you simply reduce the speed to the motors, then the robot will slam to the floor.

When a force pushes the robot, the speed limiter kicks in because it detects more movement than expected; the software quickly applies more force toward the external force's direction to counteract it. This might actually mean tilting a bit to the opposite side. We must also be careful because, if

the force suddenly stops acting, then we might get a slingshot effect that needs to be quickly dampened. If it is not, we will tip!

All these concepts might sound a little confusing, but it's actually simpler than it sounds. Picture the software in terms of subsystems. There is the balancing function that simply balances the unit; there is the speed limiting module that — according to the current status and mode — will set its limits and influence the final torque decision; there is the emergency checking module that will keep track of the maximum angle permitted and the maximum torque allowed; finally, there is the navigation module that — with the command of an external input — will set the speed and direction that the unit will move. Each of these subsystems is so simple that it can fit into a single function. At first, you should only focus on the balance and emergency stop parts, then start building from there.

Once your unit is in balance, then it's time to smooth it out and create some more efficient algorithms. Once tweaked, you should share it with the world — including me!

The turn function goes something like this:

```
PotPostion = InAdc(PotPin) – 512
SteerValue = PotPosition/Atenuator
```

where InAdc is the function that returns a value between 0 and 1,023 (10 bit resolution) from PotPin and 512 is the center of the pot. We obtain a signed value from – 512 to +512. Atenuator is the variable that will limit how fast the unit will turn. (Note that the value of 512 depends on the resolution of your ADC.)

Now for the actual turning act:

```
LeftMotor = (Torque + SteerValue)
RightMotor = (Torque – SteerValue)
```

This way, one motor will go faster, causing the unit to turn.

It's also a good practice (and one that I have not implemented yet) to keep track of the battery charge. If the batteries get low, they might not have enough power to keep the unit from falling or tipping. Temperature is another consideration. If you are loading considerable weight for the size of your PWM module/motors/batteries, the components will tend to get hot, but, if you are in an optimal range of weight and size, it will amaze you to see how efficient the concept is.

How smooth the robot will be is determined by many factors. Like I said, attention to detail is what makes it smooth and perfect. The motors I used, the open loop PWM module, the batteries, the shell, and — most importantly — the code I wrote are all very far from perfect, but the sum of all these little details gives my robot a personality. This personality can get much better with more precise math and code, but it will reach a limit where the individual components (like the motors, gears, and other physical factors in the robot) will not allow it to improve until they are modified.

What I learned is that, with these kind of materials and "perfection levels," the robot will be very smooth and efficient. What makes a big difference is the code. With clever coding, you can get away with a lot of imperfections in the mechanics and — most importantly — in the sensors.

Conclusion

Making a two wheeled robot is really not too difficult. If I could do it, I know anyone can. I am no rocket scientist — I am a hobbyist who loves electronics and challenges with cool stuff. I love to make the rules around the things I use. I know that there is probably a more elegant algorithm out there to make a robot balance gracefully.

Hopefully, those who follow me will use the Internet to share and learn. If it wasn't for all of the people that I received help from, the robot would just simply not be. I also did a great deal of research and, since I am a person who likes to give as well as receive, I made a journal to keep track of the milestones, progress, and resources I used in the project. It is available, along with the source code for experiments, on my website (**www.fusionglobal.net**); the source code is also available from *SERVO Magazine*'s website (**www.servomagazine.com**). However, in the spirit of my own work, I ask that if you have some improvement on this project, fix whatever you think is wrong and let me know about it. If you have a suggestion, please don't hesitate to contribute. If you make yours better, you should share whatever you do with the world to make a circle of improvements.

My goal is that — in the future — two wheeled, balancing robots will be as common as three and four wheeled robots and the knowledge we find now will make sure that this is no longer a challenge at all, but something taken for granted and easily incorporated into any project. Right now, the project itself is the balancing robot, but, soon, the project will be focused on another goal and the balancing part will be as simple as adding the motors and tuning them. **SV**

ABOUT THE AUTHOR

Francisco is currently a record label executive and a recording studio owner; he has been involved in robotics since he was 12 years old and started his first company devoted to laser rentals for entertainment. As a robot enthusiast, he has made several robotics projects, including artificial intelligence research and vision systems for object recognition. Currently, he is working on formalizing a company devoted to technology that will focus on stabilizing and automating an unmanned model helicopter for surveillance, entertainment, and civilian applications. His company will also sell development and education boards, as well as sensor boards for the open source community. You can visit his Spider balancing robot website at: **www.fusionglobal.net** and his personal website at: **www.lobolabs.com**

MENAGERIE

Send us a high-res picture of your robot with a few descriptive sentences and we'll make you famous. Well, mostly. menagerie@servomagazine.com

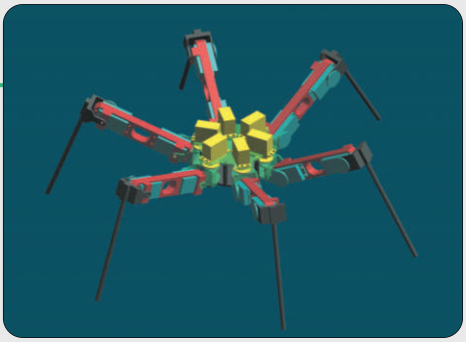
The Machine Man Band

John Rigg, RobotHut

As for the Machine Man band, I have it programmed with over 140 songs right now. I use Cake Walk 9 to do the MIDI programming for the robots. There are 100 electro-mechanical devices all under computer MIDI control, and I plan to add more some day.

It took me about seven months to learn how to make the 48 organ pipes and the electric valves to control them. I used PVC pipe, aluminum, and resin parts that I designed to make them from scratch. The second robot with the xylophone, ride cymbal, and sleigh bells took about four months to design and build.

robothut@yahoo.com

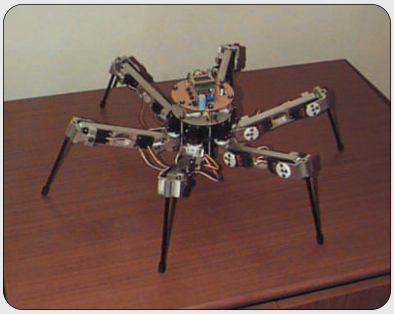


Hexapod

Carl Kalkhof, Mentor, OH

The hexapod has six legs with three degrees of freedom per leg for a total of 18 motors that need to be driven by the control system. The "brain" is a Parallax BS-2e microcontroller because it has enough memory for this application. The servo driver I used was the PicoPic because it could drive 20 servos — and was inexpensive. The PicoPic interfaces to the BS-2e through a serial line and is very easy to use. For the human interface, I found a small LCD on eBay that also interfaces to the BS-2e easily. This LCD and some push buttons allow me to access different modes of operation that have been pre-programmed in the Hexapod.

The hardest part of the whole project was finding suitable servos. I had a certain "look" in mind for this robot and wanted long, thin legs. I found that retract servos fit the dimensions, but only came in "limit to limit" operation — without proportional control. My solution was an expensive one: I used the retract servos, but substituted the non-proportional controls with micro servo controls (the only controls that would fit in the retract servo casing). This meant that I had to buy a retract servo, a potentiometer, and a micro servo for each degree of freedom.



The end result was a neat configuration that has good torque at the joints. The legs may have been the hardest consideration, but the most important one was the battery selection; I used a 7.4 V LiPoly (lithium polymer) to cut down the weight. These batteries have just recently become popular with the R/C guys and, without them, I doubt that my robot would be able to stand up! A low dropout 5 V regulator was used for the Stamp and PicoPic electronics, while three diodes in series (to drop around 2.5 V) — connected to the batteries directly — supplied the 6 V for the servos.

ckalkhof@comcast.net

New Products

ACCESSORIES

All the Battery Defender Features You've Come to Know and Love

Charge like a professional and be proactive by stopping and reversing sulfation on contact. Battery Defender Pro is a unique charger that can blast battery-destroying sulfation off your battery plates to promote longer battery life.

- **US Patent Pending**
- **De-sulfation of battery**: The charger will remove loose sulfation and increase the battery life. (Hard sulfation cannot be reversed.)
- **Small size and light weight**: Portability is a key feature. The charger can be installed onboard.
- **One charger for any type of lead acid battery (no switch)**: The Battery Defender Pro charger will charge any type or brand of lead acid battery fully — gel, sealed, or dry. The charger will adjust itself as the battery gets old.
- **Short circuit protection**: If you touch the output Prove and -ve wires (with AC ON), there will be no failure (no fuse will blow).
- **Reverse polarity protection**: If you connect the battery in reverse, there will be no failure (no fuse will blow). The charger will simply not work. When the problem is corrected, the charger will operate normally.
- **No current drain**: If the AC is OFF and the charger is connected to the battery, there will be no drain of current from the battery.
- **Low voltage start**: The Battery Defender Pro charger is the only charger that will start charging from 0.5 volts. This means a very deeply discharged battery can be charged.
- **Maintain full charge and no overcharging**: The Battery Defender Pro charger can be left connected to the battery indefinitely without harming (overcharging) the battery.
- **Super fast connect/release harness**: Includes clamp harness and lug harness with 7.5 amp fuse. Both include weather pack connections.
- **LED Indicator**: Yellow = charging; Green = charged.
- **UL**, ULc (Canada), T-mark (Japan), TUV, and CE (electrical EN 60335-1, charger EN 60335-2-29, EMC — listed above) safety approvals.
- **Soneil replacement**.
- **DC output**: 12 volt 3 amps.
- **Input**: 90-130 volts ~ 1 amp max. 60 Hz.
- **One year warranty.**

For further information, please contact:

Battery Mart
1 Battery Dr.
Winchester, VA 22601
Email: **sales@BatteryMart.com**
Website: **www.BatteryMart.com**

Circle #73 on the Reader Service Card.

CONTROLLERS & PROCESSORS

Smart Dual Channel, 240 Amp DC Motor Controller with Optical Encoder Input Targets Mobile Robots

Roboteq, Inc., now offers a micro-computer-based dual channel DC motor controller capable of directly driving up to 120 amps on each channel at up to 40 volts. The AX2850 is targeted at designers of mobile robotic vehicles including Automatic Guided Vehicles (AGV), Underwater Remote Operated Vehicles (ROVs), and mobile robots for exploration, hazardous material handling, and military and surveillance applications.

The controller accepts commands from either standard R/C radio for simple remote controlled robot applications or serial port interface. Using the serial port, the AX2850 can be used to design semi or fully autonomous robots by connecting it to single board

computers, wireless modems, or wireless LAN adapters.

The controller's two channels can be operated independently or combined to set the direction and rotation of a vehicle by coordinating the motion on each side of the vehicle. The motors may be operated in open or closed loop speed mode. The AX2850 includes inputs for two Quadrature Encoders up to 250 kHz and four limit switches for precise speed and odometer measurement.

The AX2850 features intelligent current sensing and controlling that will automatically limit each channel's power output to 120 A for the time typically required to accelerate or stop a robot. If the motor's current draw remains excessive after that time (as in the case of stalled motor or other unusual loading), the controller will gradually reduce the power to user-selected values.

The controller supports a long list of features, including analog and digital I/Os for accessories and sensors, thermal protection, programmable acceleration, input command watchdog, and non-volatile storage of configuration parameters.

The AX2850 can be reprogrammed in the field with the latest features by downloading new operating software from Roboteq's website.

The AX2850 is built into a robust, extruded aluminum case, which also serves as a heatsink for its output power stage. The large fin area ensures sufficient heat dissipation for operation without a fan in most applications.

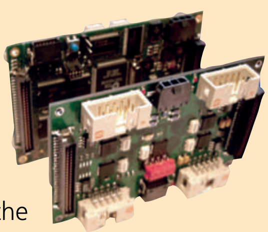
The AX2850 is available now at $620.00 in single quantities, complete with cable and PC-based configuration software. Product information, application examples, and software can be downloaded from the company's website.

For further information, please contact:

Roboteq
8180 E. Del Plomo Dr.
Scottsdale, AZ 85258
Tel: **602 • 617 • 3931**
Website: **www.roboteq.com**

Circle #45 on the Reader Service Card.

Korebot Board



KoreBot fits the needs of OEM developers of handheld devices. The size of a credit card, KoreBot is a powerful embedded platform based on the Intel® PXA-255 400 MHz processor with XScale® technology. It has 64 Mb SDRAM and 32 Mb Flash. There are two Compact Flash Type I sockets, two USB host ports, a KB-250 interface, and a JTAG debug port. The image shows a Korebot (in back) along with a Koremotor extension (in front).

KoreBot extends the same type of functionality and performance found in the latest generation of consumer PDA devices to the industrial market. Ready-to-run, extremely compact, yet flexible enough for expansion and customization for customer application needs, KoreBot provides the starting point for new designs and comes complete with all the hardware and software needed to start development — including a Linux operating system.

KoreBot is an Internet-ready appliance with a built-in web browser and server when using a pre-configured Linux operating system. With its load of I/O functions, XScale can be rapidly configured to be a data entry terminal, video player, or control system. Applications range from environmental monitoring to factory automation.

KoreBot allows quick and easy advanced application development thanks to GNU/Linux programming tools and SysQuake (developed by Calerga, Switzerland). SysQuake provides optimized numerical computation and interactive scientific visualization and can facilitate rapid and optimal robot control or robot vision software design. Visualization requires a remote host (PC or Mac) via a direct or wireless IP connection. (SysQuake for Korebot is sold separately.) K-Team provides a Linux embedded operating system and a cross-compilation tool chain for a Linux workstation to compile code on a PC. A PC-developed application can be transferred to KoreBot's flash or, by using IP over USB, code can be compiled on a workstation to be tested on KoreBot from a mounted filesystem.

KoreBot was originally designed to increase performance and connectivity while offering low power consumption for K-Team's Koala mobile robot. With Korebot's large range of input power (3-30 V) and its 10 W embedded power regulation, it is particularly well-suited for upgrading an existing robot or prototyping a new robot — from student competitions to advanced research.

KoreBot provides an expansion bus — the KB-250 Interface — dedicated to robotics extensions. The KB-250 interface is an open-standard interface to expand the functionality of an existing ARM base platform in an easy and fast way. In addition to using a full line of off-the-shelf KB-250 modules, users can also design their own modules. KB-250 modules can be used singly or stacked together to give the needed functionality. KoreMotor is the first KB-250 module developed to be used with Korebot. Other extensions are now available or will be available soon.

For further information, please contact:

Road Narrows, LLC
1151 Eagle Dr., #140
Loveland, CO 80537
Email: **oneway@roadnarrowsrobotics.com**
Website: **www.roadnarrowsrobotics.com**

Circle #67 on the Reader Service Card.

The LC8 AC Servo Motor Controller

SMC Corporation — the world's largest manufacturer of pneumatic automation products — has announced the release

of the LC8 AC Servo Motor Controller. The LC8 is a servo amplifier/controller for the LJ, LG, and LTF series actuators and provides high level features at mid-level pricing.

Up to seven LC8s can be slaved together for control from a single PLC, which can execute 117 steps that can command any or all of the actuators to move to a specific position, move a specific distance, or apply a specific amount of torque. The position accuracy is ±0.005 mm to ±0.025 mm, depending on the feed screw lead of the actuator. There are also five palletizing steps that can use the same command from the PLC to move two axes sequentially through pallets of up to 10,000 rows by 10,000 columns. The simple method for configuring and running the palletizing features has been awarded a US patent.

The LC8 comes with easy-to-use human/machine interface software for configuring and testing the controllers. This HMI software includes features for "teaching" positions by moving the actuators either manually or under power. All of the configuration data can be stored in files so that a setup for one machine can be easily duplicated on many others.

The LC8 is compact (141 mm x 75 mm x 130 mm), weighs less than 1 kg, and is available for 100-15 VAC or 200-220 VAC power and either 50, 100, or 200 watt motors.SMC Corporation manufactures high quality electric actuators and controllers for industrial automation. SMC has a total of 14 production facilities, three of which are in North America.

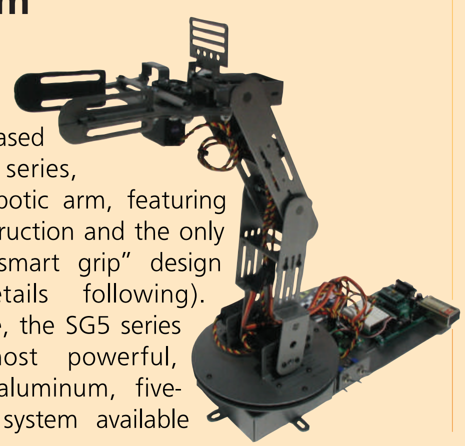
For further information, please contact:

SMC Corporation of America
14191 Myford Rd.
Tustin, CA 92780
Tel: **714•669•0941**
Website: **www.smcusa.com**

Circle #29 on the Reader Service Card.

ROBOT KITS

SG5-UT (Ultra Torque) Series Robotic Arm System



CrustCrawler, Inc., has released its new SG5-UT series, load balanced, robotic arm, featuring all-aluminum construction and the only fully-expandable "smart grip" design components (details following). Feature for feature, the SG5 series presents the most powerful, sophisticated, all-aluminum, five-axis robotic arm system available today.

- All parts are precision CNC machined from .063 gauge, 5,052 brushed, sheet aluminum.and are anodized to a smooth, scratch-resistant, graphite finish using a type II anodizing process (the hardest finish possible, next to military spec type III anodizing).
- All servo pivot points use integrated pem stud pressed pivot points, not taped or glued.
- Integrated pem nuts ease construction.
- The arm incorporates a custom engineered, counter balanced retract system that effectively ensures maximum lifting capability and efficient servo power use during operation and at rest.
- Pass through holes and slots are strategically located throughout the arm assembly for convenient wire routing.
- Two integrated SPST switches provide convenient power routing to servos and supporting electronics.
- Three integrated mounting tabs offer convenient attachment to your robotic platform.
- The arm accommodates all of the Parallax micro-controller boards.

We developed the critical design of the gripper assembly to include the following critical design features:

• The gripper assembly contains integrated, adjustable electronics that are located above the gripper assembly to accommodate an array of CCD cameras, infrared sensors, and other sensing electronics. The gripper contains four integrated slots to accommodate multiple sensing optoelectronics components. The ends of the gripper are rounded for an even gripping surface. The gripper drive system consists of a high resolution, 60 tooth, heavy duty, resin geartrain driven by a high torque Hi-Tec HS 475 servo.

• The rounded gripper ends can conveniently accommodate the "Flexiforce" pressure sensor for precise gripper pressure measurement and control (sensor not included). A 1/16" volara, cross-linked polyethylene foam is used to line the inside of the gripper surface for maximum grip adhesion. The inside width of the gripper is 3.25 x 3.25 x 1.125 inches and the arm is 17.75 inches high.It can rotate 180° on its base and lift 14.23 oz.

• The arm has three Hi-Tec HS-475 HB high torque servos for the rotating base, wrist, and gripper and three Hi-Tec HS-675MG (metal gear) ultra torque servos for the shoulder and elbow drive systems.The arm uses the Parallax Board of Education (BOE), BASIC Stamp 2 (BS2), and an all-new, powerful, 16 servo, Parallax Servo Controller (PSC).

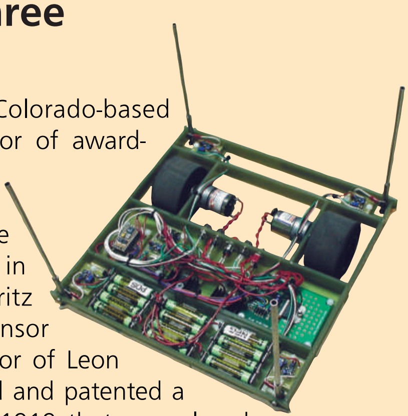
For further information, please contact:

CrustCrawler, Inc.
817 S. Capistrano Dr.
Gilbert, AZ 85233
Tel: **480•577•5557**
Email: **sales@crustcrawler.com**
Website: **www.crustcrawler.com**

Circle #94 on the Reader Service Card.

SENSORS

First Innovation in Robotic Sensors in Three Years!



Terry Fritz — a Colorado-based engineer and creator of award-winning robots — has designed the first new sensor to come on the market for robotics in several years. Mr. Fritz calls the sensor ThereminVision in honor of Leon Thermin, who invented and patented a musical instrument in 1919 that was played by a musician's hands moving near a pair of antennae.

Thereminvison II's advantage over current technology — infrared sensors — is that, with properly places antennae, you can have a full 360° detection zone.The principle of the sensor is based upon the fact that there is a very weak electromagnetic field that surrounds an antenna. When a conductor enters the electromagnetic field, it changes the field's capacity in a measurable way. This conductor can be a hand or a metal object — such as a robot. Almost all objects have some detectable capacitance that changes the electromagnetic field.

ThereminVision II is an inexpensive kit that, when assembled (requires soldering), allows experimenters and robotics enthusiasts to detect when objects approach. The output from the kit is wired to a microprocessor, which is not supplied with the kit. The kit consists of four sensor boards and a processor board. The sensor boards — when placed at the four corners of a robot and connected to antennae — form an electromagnetic field around the robot that is a zone of detection for objects.

A microprocessor connects to the kit's processor board and polls the sensors. As an object approaches the sensor's antenna, it will send increasingly smaller bits to the microprocessor. The antenna is *not* included, since the kit builder needs to incorporate it into the design. Design guidelines and sample programs are included in the manual.

Features of the kit include:

- Four sensors. Each sensor weighs 3.7 grams, draws far less than 1 mA at 5 volts, and is less than 1 x 1.25 inches.
- The processor — a separate board — weighs 10.7 grams, at less than 1mA at 5 volts, and is 1.3 x 2.4 inches.
- There are two digital control lines and one output signal line connected to a microprocessor which measures pulse width. An optional control line turns the sensors off.
- Power can easily be drawn off the +5 volts of most microprocessor supplies.
- Detection range is proportional to the antenna and object surface areas. See manual for details. Programming could randomly move the robot until an object is detected.
- A downloadable manual is available.

A demonstration robot and a kit can be seen at: **www.robotlandinc.com/tvision.htm** The kit is $50.00 US (plus S & H).

A discussion group devoted to the Thereminvision II kit is at :**http://groups.yahoo.com/group/thereminvision/**

For further information, please contact:

Robotland, Inc.
345 E. Camelback Rd.
Phoenix, AZ 85012
Email: **info@robotlandinc.com**
Website: **www.robotlandinc.com**

Circle #50 on the Reader Service Card.

When Data Goes Bad — Error Detection and Correction

by Jack Buffington

Transmitting data between devices is a common task for electronic projects. When the devices are near each other and directly connected with a cable or circuit board trace, you can often ignore the possibility of data errors. If, on the other hand, the devices are separated by a long stretch of wire or — even worse — by a radio link, then you need to start thinking about ways to detect when data errors have occurred and how to correct them, if necessary.

Data errors can be caused by any number of things, but common sources are electromagnetic interference from outside sources, such as motors that are electrically noisy, other radio transmitters nearby, or by the shape of your data bits changing as they travel down a wire or are being distorted by your radio equipment.

For the purposes of this article, it will be assumed that all of the data we are referring to is in serial format. This article will discuss a few different strategies to help detect errors when they are received and then move on to strategies to recover data that is determined to be bad. Finally, it will detail how to avoid errors in the first place.

The first and simplest method that you could use to detect if there has been an error would be to simply send each byte twice. At the receiving end, you can just subtract one byte from the other. If the result is zero, then the data is likely to be good.

A slightly better method would be to send your first byte normally and the second byte with all of the bits inverted. By performing an XOR operation on the data, you will be able to detect if the data is good, as the result would be all ones. Inverting the second byte is slightly better because, if the bytes are sent one after the other and the error is caused by an intermittent short in a wire, it will tend to drive all of the bits either high or low. XORing the data will detect this sort of situation. XOR stands for eXclusive OR. The truth table for XOR looks like Figure 1.

A	B	XOR
0	0	0
0	1	1
1	0	1
1	1	0

Figure 1. Exclusive OR truth table.

	10111010
XOR	01011011
	11100001

Figure 2. An example of XOR.

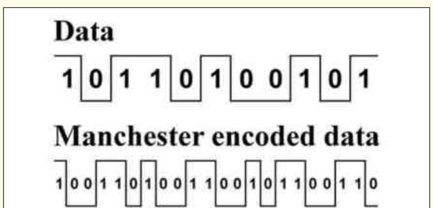


Figure 3. Manchester encoded data.

Manchester Encoding

Another simple way to assist in the detection of errors is to encode your data using Manchester encoding. Like the previous strategy, this method doubles the amount of bits that need to be sent. Manchester encoding is a method of encoding your serial data so that both the clock signal and the data are sent simultaneously.

This type of encoding gives you the flexibility to change your data rate dynamically because the receiver will be able to sync itself to the sender's clock. This can be useful in situations where you are experiencing a lot of noise.

To encode data using Manchester encoding, you will replace each bit that you are trying to send with two bits. If the data bit that you want to send is a 0, then you would send the two bits "01." If the data bit that you want to send is a 1, then you would send the two bits "10." The end result of this sort of encoding is that the signal that is sent will never stay high or low for more than the time needed for two bits. If the signal stays high or low for longer, then you will know that you are receiving an error.

Parity

A third and very common way to detect errors in your data is to use something called parity. If you are using parity, you will add an additional bit to each byte that you send. Parity can be either even or odd. The value of the parity bit

will be determined by counting the number of 1s in the byte that you want to send. For even parity, the goal is to have an even number of 1s between the byte and its parity bit.

For example, in Figure 4, the first byte has four 1s in it, so the parity bit is a 0 because there is already an even number of 1s in the byte. Odd parity is just the opposite. In odd parity, the goal is to end up with an odd number of 1s between the byte and its parity bit. Using parity will detect if an odd number of bits have been inverted, but misses cases where an even number of bits have been inverted.

Checksums

Another, slightly more complex, way to detect errors is to use something called a checksum. This method of error detection fits better with larger chunks of data than a single byte, unlike the previous examples. There are many possible ways to compute a checksum.

Here is a very basic example of how you could compute one. We will have packets of six bytes. For each six bytes of data, we will add a seventh byte of data that is the checksum byte. To calculate the value of the seventh byte, you could add together the six data bytes and discard any extra bits that overflow. The sender would send this data packet. On the receiving end, the six bytes would be added together and the result would be compared to the checksum. If the checksum and the computed result were the same, then the data would be considered to be valid.

Checksums can get more complex than this example. The next example demonstrates how to compute an Adler-32 checksum. The Adler-32 checksum is 32 bits long and is computed as two 16-bit values. The first value is simply the sum of all bytes in the packet plus 1. The second value is a little more complex. Refer to Figure 5 for a better idea of how to calculate the second value.

The second value is the sum of all of the intermediate first value results. For each step of calculating the second value, you will take the total of the previous step, plus the running sum from the first value. After the first and second values are calculated, there is one additional calculation that must be done on them — a modulus operation.

To calculate modulus, you divide the first number by the second number using integer division. The result is the remainder. For example, 7 mod 2 is 1, since 7 divided by 2 has a quotient of 3 with a remainder of 1. In the case of the Adler-32 calculation, we are calculating each value mod 65521. To send this packet, first send the actual data, followed by the first and second values of the checksum.

Cyclic Redundancy Check (CRC)

CRC error checking is the strongest type of error checking of those presented here. It works with longer sequences of data, but also takes more processor power than the previous examples. Like any algorithm that is often used, CRC has other, more highly optimized forms that can be somewhat cryptic to follow.

Like the other error detection methods discussed here, CRC sticks a small piece of data onto the end of each chunk of data that is sent. The CRC algorithm works in a similar way to binary division, which was discussed in May's column.

There are three differences between binary division and the CRC calculation. The first is that, instead of using subtracts at each step of the calculation, CRC uses the XOR operator. The second difference between binary division and the CRC calculation is that, at each step of the solution, the decision whether or not to do the XOR operation is made contingent upon if the most significant bit of the partial remainder is a 1 or a 0. If the most significant bit is a 1, then the XOR operation is performed. The third difference between binary division and the CRC calculation is that, before the CRC calculation, a certain number of zeros is appended to the end of the dividend. The number of zeros appended is the number of bits in the divisor minus one.

Take a look at Figure 6 for a better understanding. The important part of the CRC calculation is the remainder. This is the part that is tacked onto the end of the CRC packet. In Figure 6, the digits in green are the original message. The blue zeros are added on to assist in the calculation. The orange digits are the remainder that replaces the blue zeros in the final packet that is transmitted. In Figure 6, the message being sent would be 1010 1011 1101.

There is one more part of CRC that hasn't been discussed yet. This is the divisor that is used. The specific divisor used has a big influence on how likely it is that the CRC calculation on the receiver side will detect a given error. Figure 7 lists some commonly used ones. This article will not go into why each of these divisors is better than others, but rest assured that some really smart math people out there have some good reasons and statistics to back them up!

Once all of this is done, the packet is

Even Parity		Odd Parity	
10110100	0	10110100	1
00101100	1	00101100	0

Figure 4. Even and odd parity.

Character	ASCII	First value	Second value
S	83	1 + 83 = 84	0 + 84 = 84
E	69	84 + 69 = 153	84 + 153 = 237
R	82	153 + 82 = 235	237 + 235 = 472
V	86	235 + 86 = 321	472 + 321 = 793
O	79	321 + 79 = 400	793 + 400 = 1,193

Figure 5. Calculating an Adler-32 checksum.

```
          1100 0101
11001 | 1010 1011 0000
       1100 1
        110 00
        110 01
         00 011
         00 000
          0 0111
          0 0000
            0111 0
            0000 0
             111 00
             110 01
              01 010
              00 000
               1 0100
               1 1001
                 1101
```

Figure 6. An example of a CRC calculation.

transmitted to the receiver, which can do one of two things. The first is to strip off the CRC result and perform the same calculation to see if it arrives at the same result. If so, then the data is good. The other thing that it can do is to just do the CRC division with the CRC value still attached and see if it arrives at an answer of 0. If so, then the data is good.

Error Correction

Sometimes, it is important for a system to be able to correct data that it receives all by itself. This might be the case if the data can only be transmitted once due to time constraints or memory limitations. Another situation might be if the lag time between transmission and reception is large and a request for a repeat of the data would take quite a long time. The Mars Rovers are likely employing some sort of error correction because the delay between a request for a repeat of data and the reception of that data is around seven minutes!

One simple way to correct for errors is to just have a best two of three situation, where each byte is sent three times. If two of the bytes match, then it is likely that they have not been corrupted. That is pretty simplistic and also requires that the amount of data sent be tripled.

Block Sum Checking

Another strategy that requires sending less extra data is Block Sum Checking. This is essentially a two-dimensional parity check where an array is made out of the data bytes and parity is calculated for each row and column. This method allows you to correct for any single-bit error and to detect most other errors.

Figure 8 shows eight bytes aligned in rows. Parity is calculated for each row and column. If this data is sent to the receiver and one bit is inverted, then the parity values will not match and we will be able to figure out which bit changed.

In Figure 9, we can calculate the parity for each row and column to find that the parity values that are shaded blue do not match the value we are expecting. Since we have exactly one bad column parity and one bad row parity, we can deduce that the bit shaded red was the bit that inverted and we can then correct it.

The Block Sum strategy has one Achilles heel. If there happen to be four bit errors arranged in the pattern shown in Figure 10, then this strategy will not detect it. As you can see, the four bit errors are arranged in a square. Since parity cannot detect even numbers of bit errors and each row and column containing errors has two errors, Block Sum Checking will miss these errors.

So which method of error detection or correction is right for you? Things to think about are how important it is that the data is received flawlessly, how much processing power you have available, and how much RAM you can devote to error checking.

If you need to ensure that data gets to its destination flawlessly, then you might choose CRC error detection. This can detect any single error and multiple errors that are within the number of bits in the divisor, –1 of each other. If multiple errors are spaced farther than the number of bits in the divisor, –1 it is still over 99.9% effective at detecting errors.

The Adler-32 strategy of error detection is almost as effective, but can be calculated even faster. If you just want some simple protection for your data and don't want to do much calculation or send a lot of bits, then using parity will catch errors 50% of the time. Sending duplicate data in two bytes reduces your chance of an error getting through to less than .4%, but does require double the bandwidth.

1100000001111	CRC-12
11000000000000101	CRC-16
10001000000100001	X25 standard or CCRC-CCITT
100000100110000010001110110110111	CRC-32

Figure 7. Common CRC divisors.

Error Prevention

In the film industry, a common thing that you'll hear on a set is, "we'll fix it in post-production." This seems like such a strange thing to hear when doing it right in the first place is often easier. This article wouldn't be complete without talking about error prevention. Why deal with a lot of errors if you can prevent them from happening in the first place? Here are a few strategies that you can take into consideration when designing something that might send or receive data that could be corrupted.

If you are sending your data through wires, opting for a wire that has shielding around the conductors may reduce the amount of errors that you receive. If you are transmitting your data through a long length of cable, you might consider using a differential transmission mode instead of a single-ended one.

An example of a single-ended protocol is what comes out of the RS232 port on the back of your

10110110	1
11011100	1
01110111	0
00010010	0
11110000	0
10101010	0
01010101	0
10001111	1
10001111	

Figure 8. Block Sum using odd parity for eight bytes.

10110110	1
11011100	1
01110111	0
00000010	0
11110000	0
10101010	0
01010101	0
10001111	1
10001111	

Figure 9. An error that can be corrected.

10110110	1
10011000	1
01110111	0
00010010	0
10110100	0
10101010	0
01010101	0
10001111	1
10001111	

Figure 10. Block Sum can't detect this error.

computer. This port uses one wire as ground and the other as data. Differential transmission — as that used in RS485 — sends the data through two wires that are either driven high or to ground, but are always opposite each other. On the receiver end, a comparator extracts the data.

Another common problem with data transmission is crosstalk. This can even affect short traces on a circuit board. Crosstalk is where one wire can pick up the signals from a nearby wire, even though there is no direct connection. If it is possible, separate wires that carry different data as best as you can. It also helps to put ground wires between signal wires if you happen to be sending the data through a ribbon cable.

One final thing to think about if you are sending data through wires is the possibility that these wires might act as antennae and pick up radio waves from transmitters that are close by. This can be an infuriating problem because it can be sporadic. A way to solve this is to put a low pass filter onto your data line. It is highly likely that the radio waves will be a much higher frequency than your data is and can be easily filtered out while leaving your data pulses relatively intact.

If the method that you are using to send your data is wireless, then there are a few things to think about, as well. The first thing would be to determine if you really need wireless capability. Wireless communications are often noisier than direct connections. Picking the frequency that you will be using is important. If you choose a frequency that has a lot of use, you are more likely to encounter interference.

The method that you use to send your data can affect its reliability. Amplitude Modulation (AM) is prone to noise interference, but it is inexpensive to add to your project. Frequency Modulation (FM) is less prone to noise, but is more expensive.

You may decide to use a premanufactured radio module. Not all radio modules are equal, so you may want to test out a few different ones to find what fits best with your application. One other thing to consider when selecting a radio module is that, even though the specification for your radio module may say that it can transmit at a certain data rate, the data rate specified may be under ideal circumstances. By lowering the speed that you send your data, you may find that the number of errors you encounter decreases.

Error detection and recovery is something that you will likely need to deal with at some point if you work long enough with electronics. Hopefully, this article has given you some insights into how to ensure that your data is received reliably and, if it is not, how you can detect bad data before it is too late. **SV**

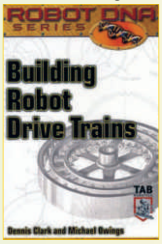
Author Bio

When not writing for *SERVO Magazine*, Jack runs Buffington Effects, a company that designs and builds animatronics and motion control devices for the entertainment industry. Check out his website at **www.BuffingtonFX.com**

Mind Candy For Today's Roboticist

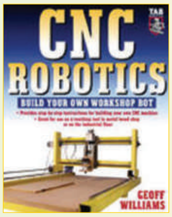
The *SERVO* Bookstore

Building Robot Drive Trains
by Dennis Clark / Michael Owings



This essential title in McGraw-Hill's *Robot DNA Series* is just what robotics hobbyists need to build an effective drive train using inexpensive, off-the-shelf parts. Leaving heavy-duty "tech speak" behind, the authors focus on the actual concepts and applications necessary to build — and understand — these critical, force-conveying systems. **$24.95**

CNC Robotics
by Geoff Williams



Now, for the first time, you can get complete directions for building a CNC workshop bot for a total cost of around $1,500.00. *CNC Robotics* gives you step-by-step, illustrated directions for designing, constructing, and testing a fully functional CNC robot that saves you 80 percent of the price of an off-the-shelf bot and can be customized to suit your purposes exactly, because you designed it. **$34.95**

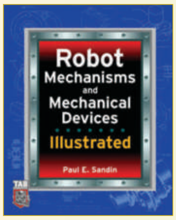
PIC Robotics: A Beginner's Guide to Robotics Projects Using the PIC Micro
by John Iovine

Here's everything the robotics hobbyist needs to harness the power of the PICMicro MCU! In this heavily-illustrated resource, the author provides plans and complete parts lists for 11 easy-to-build robots — each with a PICMicro brain. The expertly written coverage of the PIC Basic Computer makes programming a snap — and lots of fun. **$19.95**

Robots, Androids, and Animatrons, Second Edition
by John Iovine

There's never been a better time to explore the world of the nearly human. You get everything you need to create 12 exciting robotic projects using off-the-shelf products and workshop-built devices, including a complete parts list. Also ideal for anyone interested in electronic and motion control, this cult classic gives you the building blocks you need to go practically anywhere in robotics. **$19.95**

Robot Mechanisms and Mechanical Devices Illustrated
by Paul Sandin



Both hobbyists and professionals will treasure this unique and distinctive sourcebook — the most thorough — and thoroughly explained — compendium of robot mechanisms and devices ever assembled. Written and illustrated specifically for people fascinated with mobile robots, *Robot Mechanisms and Mechanical Devices Illustrated* offers a one-stop source of everything needed for the mechanical design of state-of-the-art mobile 'bots. **$39.95**

Robot Builder's Bonanza
by Gordon McComb

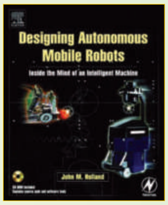
Robot Builder's Bonanza is a major revision of the bestselling bible of amateur robot building — packed with the latest in servo motor technology, microcontrolled robots, remote control, LEGO Mindstorms Kits, and other commercial kits. It gives electronics hobbyists fully illustrated plans for 11 complete robots, as well as all-new coverage of Robotix-based robots, LEGO Technic-based robots, Functionoids with LEGO Mindstorms, and location and motorized systems with servo motors. **$24.95**

Robot Builder's Sourcebook
by Gordon McComb

Fascinated by the world of robotics, but don't know how to tap into the incredible amount of information available on the subject? Clueless as to locating specific information on robotics? Want the names, addresses, phone numbers, and websites of companies that can supply the exact part, plan, kit, building material, programming language, operating system, computer system, or publication you've been searching for? Turn to the *Robot Builder's Sourcebook* — a unique clearinghouse of information that will open 2,500+ new doors and spark almost as many new ideas. **$24.95**

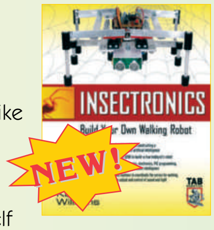
We accept VISA, MC, AMEX, and DISCOVER
Prices do not include shipping and may be subject to change.

Designing Autonomous Mobile Robots
by John Holland



Designing Autonomous Mobile Robots introduces the reader to the fundamental concepts of this complex field. The author addresses all the pertinent topics of the electronic hardware and software of mobile robot design, with particular emphasis on the more difficult problems of control, navigation, and sensor interfacing. Its state-of-the-art treatment of core concepts in mobile robotics helps and challenges readers to explore new avenues in this exciting field. The accompanying CD-ROM provides software routines for the examples cited, as well as an electronic version of the text. **$49.95**

Insectronics
by Karl Williams



This complete project book delivers all the step-by-step plans you need to construct your own six-legged, insect-like robot that walks and actually responds to its environment. By using inexpensive, off-the-shelf parts, hobbyists can "build a better bug" and have loads of fun honing their knowledge of mechanical construction, programming, microcontroller use, and artificial intelligence. **$19.95**

Amphibionics
by Karl Williams

If you're a robotics hobbyist with a flair for creativity, here's your opportunity to join the revolution and advance robotic evolution. This work provides the hobbyist with the detailed mechanical, electronic, and PIC microcontroller knowledge needed to build and program snake, frog, turtle, and alligator robots. It focuses on the construction of each robot in detail and then explores the world of slithering, jumping, swimming, and walking robots — and the artificial intelligence needed with these platforms. Packed with insight and a wealth of informative illustrations, *Amphibionics* focuses on construction details and explores the artificial intelligence needed to make these specialized movements happen. **$19.95**

Our resident expert on all things robotic is merely an Email away.
roboto@servomagazine.com

Tap into the sum of *all human knowledge* and get your questions answered here! From software algorithms to material selection, Mr. Roboto strives to meet you where you are — and what more would you expect from a complex service droid?

ASK MR. ROBOTO

by
Pete Miles

Q. I have read with interest your reply in the May edition of *SERVO* regarding inexpensive microcontrollers. An alternative your reader might like to consider is the PICAXE system. These are PIC chips with bootloaders installed. They are programmed using a free Basic editor and the programming circuit consists of nothing more than two resistors. They are surprisingly powerful with the higher end products, sporting up to eight servos or two pwm lines operating in the background. Details can be found at **www.rev-ed.co.uk/picaxe/**

I have no connection with this company other than that of a satisfied customer and devotee.

— James Carter
via Internet (UK)

A. Thanks for the note. After I received your Email, I went to their website to check them out. I was quite impressed with them. The Basic compiler is free, can be downloaded from their website, and has all the features you need to be able to program the microcontroller to do whatever you want it to do. All you need is a simple three wire serial programming cable (that you can purchase or build yourself) to program the PICAXE microcontrollers.

They have some excellent documentation on their website that shows you how to use their products, along with some excellent tutorials for learning how to use microcontrollers, electronics, and robots. They currently offer seven different microcontrollers with 5 to 32 lines of I/O control. The cost for the PICAXE microcontrollers is slightly more expensive than that of bare PIC microcontrollers. These prices — along with the free Basic compiler — make the PICAXE microcontroller the least expensive option for the hobbyist.

Q. I have a robot that is controlled by a laptop computer. I'm using external speakers so that I can hear the robot talk when the screen is closed. The problem that I am having is that when I power up the servos, I can hear the noise of the servos through the speakers.

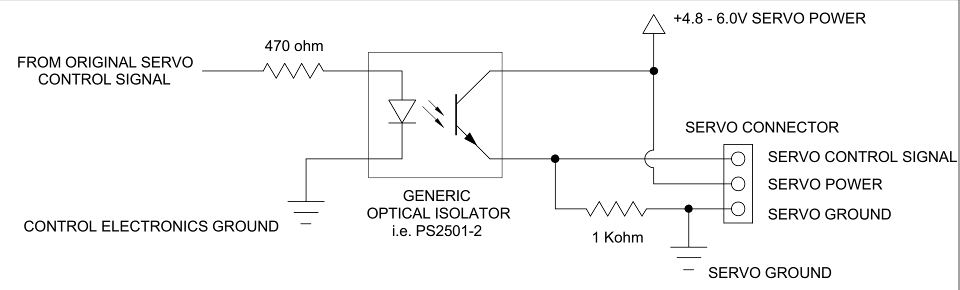
The board used to control the servos is a USB Servo II by ACS. The power jack to the servo control board also powers the speaker amp. Is there anything I can put between the jack and the amp to stop this noise? Do I have to put them on a separate power supply?

— Wone Barnwell
via Internet

A. As a general rule of thumb, electric motors should always be connected to a separate power supply, since they are notorious sources of electrical noise. When using two separate power supplies, make sure that both of the grounds are tied together, so that all the electronics are using the same ground reference, or you will see erratic behavior in the components. If you have to use the same power supply, then try adding some 0.1 μF capacitors between the positive and ground wires as close to the servos as possible.

The best way to eliminate the noise from the servos is to completely isolate them from the rest of the circuit. This also means using two separate power sources (one for the

Figure 1. Servo optical isolation circuit example.



servos and the other for the rest of the circuit) and placing optical isolators between the servos and the control signals. In this configuration, the grounds of both power supplies are not tied together.

A simple schematic for isolating the servos from the rest of the circuit is shown in Figure 1. The 470 Ω resistor is used to limit the current to the LED. The 1K resistor is used as a pull down resistor to ensure that the control signal to the servo is at zero volts when the LED is off. Any optical isolator should work in this circuit. The PS2501-2 from NEC has two separate optical isolators and works well in this application.

Using a circuit like this between each one of your servos should solve the noise problems you are having. If not, the noise is coming from a different source.

Q. I'm writing in regard to the Hexatron robot plans you featured in your November 2003 issue of *SERVO*. First of all, I realize that none of this is your problem or your fault, but I'm hoping you can help.

I'm new to robotics, but very excited about it. I decided to build the Hexatron robot — my first non-toy robotic project. For about two months now, I've been trying everything I know of to find the parts for it. This includes Emailing the author and many other people. Though I've had to go to dozens — if not hundreds — of sites, I've found all but the 4.55K potentiometer. Would you have any idea where I might find this part? I thought I had it at **www.Stampbuilder.com** but I was wrong.

The problems I have been going through to find these parts are very disheartening. I can't help but wonder how many people new to the robotics hobby have turned away due to such problems. I'm tempted to start my own robot supply business after this experience, but I can't even do that until I find a source for the remaining part I need. The author of the article — Karl Williams — did a fabulous job on the instructions and pictures. It's a shame that he did not give enough information on the parts to allow a novice hobbyist to build the robot. For instance, there are dozens of variations of semiconductors that have "2N4403 PNP general purpose" as part of the description.

Thanks for reading my gripes. If I can just find this last part, I'm sure the project will be a blast.

— Richard Alexander
Broken Arrow, OK

A. Welcome to the fun — and sometimes frustrating — world of robotics. You are not alone here; every electronic hobbyist in the world experiences the same type of frustration that you are experiencing now. One of the nice things about electronics is that every part can be substituted with a different one and perform the same function. Often, these parts are not even remotely similar to one another.

The trick is learning how to pick the right parts to substitute with the parts you can't find. Actually, there is no trick or black magic here; all that is required is to figure out what the part is being used for in the circuit. Understanding basic electronic fundamentals helps you, along with experience.

First off — if you cannot find the electronic parts at Digikey (**www.digikey.com**) or Mouser Electronics (**www.mouser.com**) — then this should tell you that they are not everyday, commonly used parts; hence, they will be more difficult to find. When it comes to searching for difficult to find semiconductors, I use FindChips.com (**www.findchips.com**). If FindChips doesn't locate them, then they will be very hard to find.

Now, let's take a look at the Hexapod circuit as an example to learn how to substitute parts. The 4.5K Ω potentiometer is connected between the +5V power supply and the ground. The potentiometer's center tap is connected directly to the analog-to-digital converter (ADC) port on the PIC. Since the ADC port is going to draw almost no current from the potentiometer, the potentiometer is going to act like a variable voltage divider. Thus, depending on the rotational position of the potentiometer, the ADC port on the PIC will see an analog voltage that will range between 0 to 5 volts. In this configuration, any potentiometer — regardless of its value — will do the same exact same thing.

The next question to answer is why use a 4.5K Ω potentiometer? First, from Ohm's law, the current that will go through the resistor will be limited to 1.1 mA (V/R). Since this is a battery-operated robot, you will want to minimize the amount of current that is being wasted going through

the resistor so that the batteries will last longer. Secondly, the 4.5K Ω potentiometers are probably what Karl Williams had in his workshop. As you already discovered, a 4.5K Ω potentiometer is not standard. The closest commonly available potentiometers are 5K Ω. Making this substitution will work just fine in the robot.

For the second part, the schematic drawing for the Hexapod shows that the 2N4403 PNP transistor is part of the H-Bridge for the motor. In this configuration, the transistor is only being used as a switch to control the current flow direction through the motors. So, the main concern for these parts is how much current the transistor can tolerate. Keep in mind that the bottom two transistors must be PNP types and the top two must be NPN types. Yes, there are a lot of companies out there that make many different versions of the 2N4403 PNP, but it really doesn't matter which one you use.

The main thing that you should look at is how much current the transistor can handle — the maximum current specification — IC max — for the transistors. The 2N4403 transistor has a typical maximum current rating of 600 mA; thus, any PNP transistor that can handle this much current can be used. Using a smaller-sized (lower current rating) transistor will most likely result in the transistor being damaged.

One of the other things that needs to be considered is the physical package size of the transistor. The 2N4403 PNP transistor comes in a TO-92 package. If you are planning on using the circuit board that is recommended for this project, then you may be limited to this type of a physical package for the transistor. If you are going to make your own circuit board, then you can consider using TO-220. This transistor is physically larger than the TO-92, but it can handle tens of amps, where as the TO-92 package is generally limited to less than 1 amp of current.

The reason I mention this is that robot motors are often stalled — either on purpose or by accident — and the stall current can be significantly higher than the normal operating current. I have the same type of motors at home and the free running current is around 500 ma, but — when the motors are stalled — the current draw is over 2 amps! The 2N4401 and 2N4403 transistors can handle this only for a short time before they pop.

I personally would use the TIP102 NPN and the TIP107 PNP transistors instead of the 2N4401 and 2N4403 transistors, since they can handle 8 amps of current. This way, I don't have to worry about stalling the motors (at least I don't have to worry about blowing up the transistors), but you won't be able to use the printed circuit board drawing that is provided in the article.

I hope this gives you some insight into how to figure out how to get parts when you can't find the ones that are specified in the drawings. Remember, electronics are fairly forgiving about using different parts that don't have the exact same specifications and, yet, still perform the same functions. This is one of the reasons why this is such an enjoyable hobby.

Q. I want to build a security robot, but I do not know how to communicate with the sensor to say, "Hey, there is something going on," and have the robot respond and check the area. I would like to know which sensor is best and how I get them to communicate.

— Anthony Avila
via Internet

A. Sensors generally have two different types of outputs — analog and digital. The digital output sensors will tell you whether there is something there or not, open or closed, on or off. The analog sensors have a variable voltage (for example, 0 to 5 volts) or variable current (for example, 4 to 20 mA). What you need to do is have your security robot monitor all of the sensors and, when one of the sensor's values changes, the robot should investigate the area where the sensor indicated an intrusion.

Security systems generally use the digital style sensors — such as magnetic reed switches (to check to see if doors or windows have been opened), motion sensors (to see if something is moving in front of the sensor), or invisible lasers (to act as optical trip wires). Motion sensors are one of the more popular types of sensors used in security systems. Once set in place, the motion sensor will detect movement by looking for sudden changes in infrared heat in front of them. When they detect something, they activate a relay that is normally connected to a light or an alarm. Motion detectors can be found at most hardware stores. In fact, all of these types of sensors can be found at hardware or home security stores.

For home security applications, you should look at the various sensors at **www.x10.com** and **www.smarthome.com** What makes the X10 devices popular is that they use a communication method that transmits through the regular 120 VAC power lines in your home or by wireless radio communications, so you don't need any additional wiring. The various X10 compatible devices have interfaces that will directly connect to a computer, so the computer can monitor and control the home. These interfaces make it much easier to interact with a robot.

Another type of sensor that could be used in a security environment is long range ultrasonic sensors. Acroname (**www.acroname.com**) sells several ultrasonic ranging systems — such as the SensComp 7000 package — that can detect objects out to a range of 35 feet. For security purposes, you would use the ultrasonic range sensors to detect a change in the range of something, but, then again, it could also be used to track something coming toward the sensor. Things that move past the sensor, but not systematically closer to it — such as a bird flying by — could be ignored, whereas, if something was continually moving closer to the sensor, this would alert the control system that something needs to be checked out by the robot.

If you are not familiar with hooking up sensors to robots or microcontrollers, I would recommend that you take a look at the *Applied Sensors* book that can be downloaded from the Parallax website (**www.parallax.com**). Hopefully, all of this information will get you on the right track. SV

Book Review

by Dr. Ralph Lorenz

SOJOURNER

An Insider's View of the Mars Pathfinder Mission

This book tells the story of the development of NASA's Sojourner Rover, which — until this year — was "the" Mars Rover. It landed on Mars in July of 1997, driving off the Pathfinder lander and capturing the public's imagination — finally something sent to another planet that didn't just sit there! However, it needed new innovations in sensors and autonomy because it was so far from Earth that control signals took 20 minutes or more to reach it.

The story begins with the early work at NASA's Jet Propulsion Laboratory (JPL) on a much larger rover, which became too big and expensive to fly as NASA's budget shrank in the late 1980s. Some clever work that led to the "rocker bogie" suspension system made it possible to have a small rover that could still climb over substantial obstacles, just as an opportunity to fly a rover to Mars came about. So, Sojourner was born. The historical account is leavened with many personal anecdotes that make it a compelling read. The tale gets into some nice technical details that *SERVO* readers will appreciate, like how they burned out the FETs on an H-bridge circuit while hot wiring a motor during tests or how tape innocently used to mark a grid on the floor looked completely different to the human eye and to the rover's infrared obstacle sensors, foxing the rover. These real world hardware problems and their solutions, along with the similar difficulties in software and in rover operation, make instructive reading.

The insights into the political maneuvering needed to make something new and unconventional happen are just as fascinating as the technical details of Sojourner and its predecessors. The Mars Pathfinder heralded the "Faster, Better, Cheaper" era in NASA and the Sojourner Rover was even more of a long shot — a technological "add-on" to a scientific mission. Clashes of personalities, the importance of getting the right people for the job, and the sheer stamina needed to solve one challenging problem after another on a tight schedule all come through in the brisk narrative.

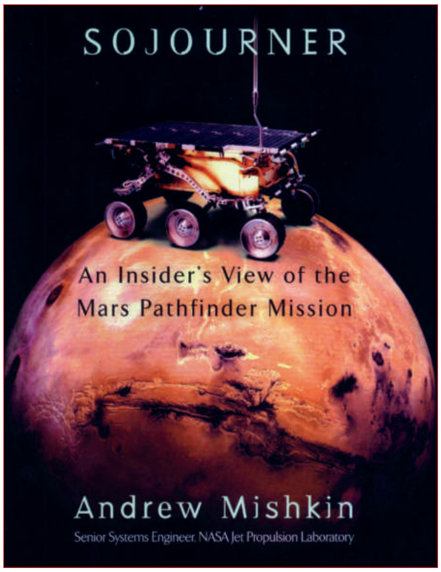
The book has a dozen or so black-and-white photographs and a list of abbreviations. No bibliography or directions for further reading are given, but an index is provided. It is a personal story by one of the main developers of the project. As such, it focuses on some aspects much more than others, notably giving woefully little discussion of the scientific results of Sojourner. Doubtlessly, other members of the project might have a different outlook on some of the exchanges described in the book or on what the most important parts of the story are, but the first-hand account is no less interesting for all that.

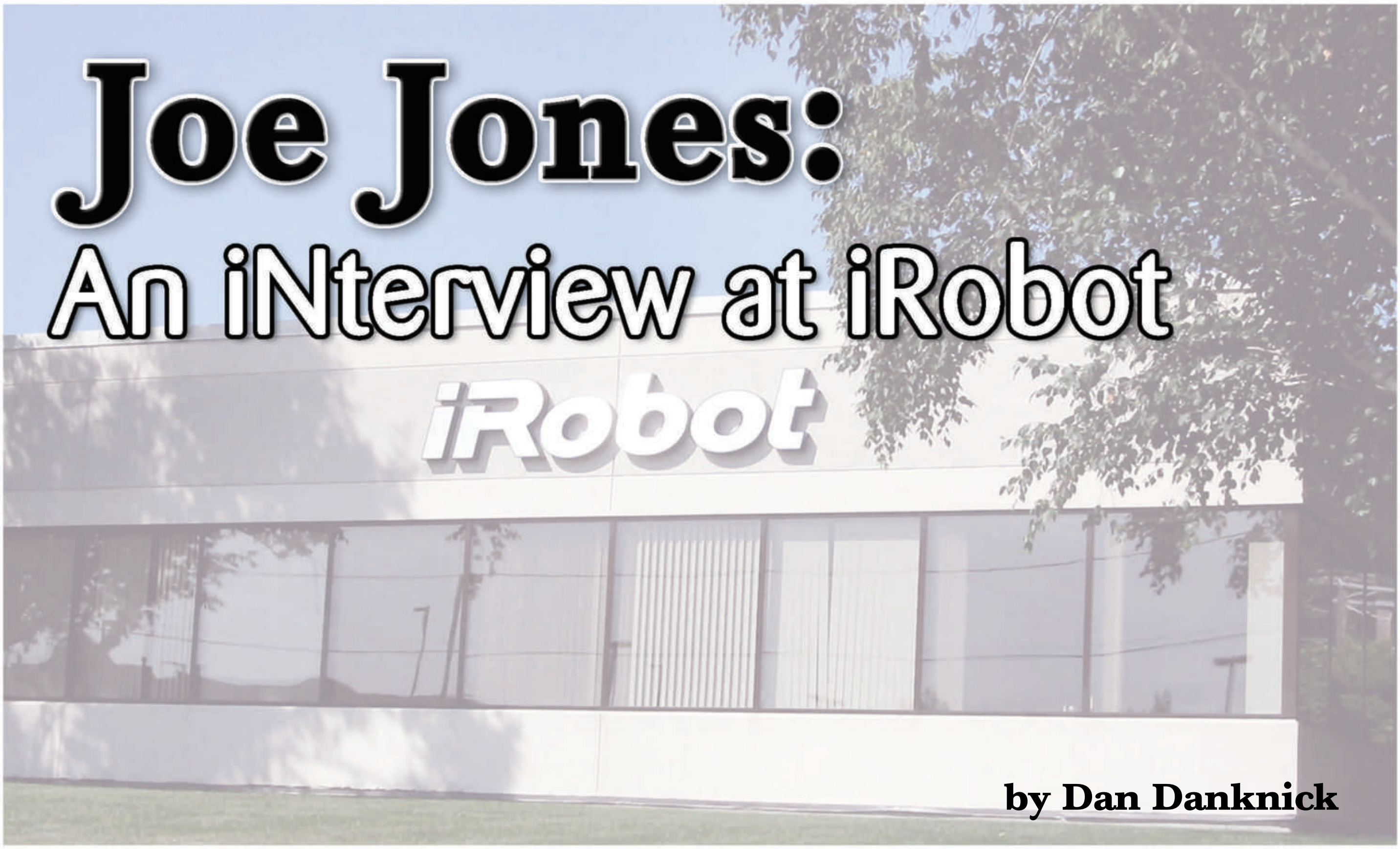
Building robots is what many of us do for fun. Mishkin gets to do it for a living and his robots drive around on Mars — a dream career for many *SERVO* readers. Mishkin shares his enthusiasm — as he notes in the book, "Damn, I love my job." I can recommend this readable account to anyone interested in space exploration and robotics. **SV**

About This Book ...

Sojourner: An Insider's View of the Mars Pathfinder Mission
by Andrew Mishkin

Hardcover 333 pages
Berkeley Publishing, 2003
List price $21.00





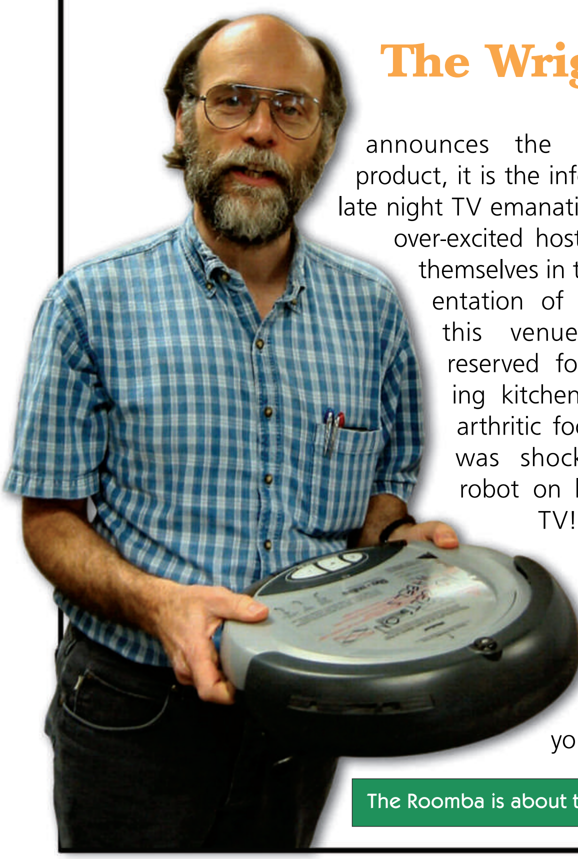
Joe Jones:
An iNterview at iRobot

by Dan Danknick

From AI to TV

If there is a cultural marker that announces the success of a product, it is the infomercial — that late night TV emanation replete with over-excited hosts tripping over themselves in the gleeful presentation of a product. As this venue is normally reserved for the time-saving kitchen appliances of arthritic football players, I was shocked to see a robot on late-night cable TV! Yes, a robot!

The Wright brothers used the shape of a wing,

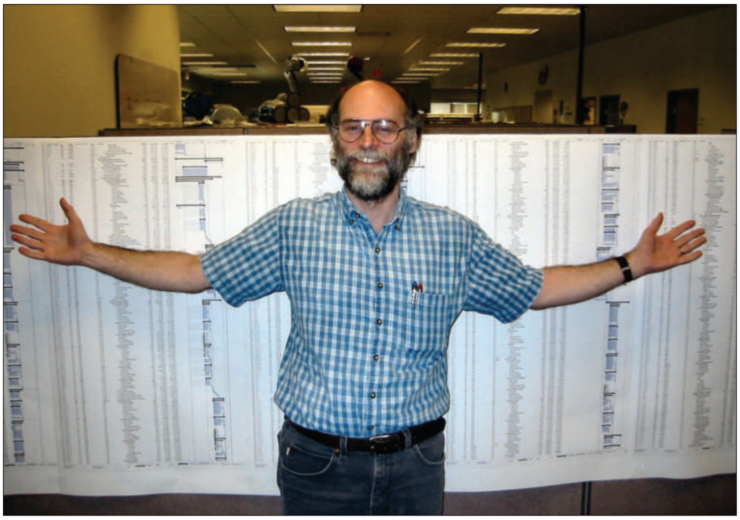


The Roomba is about the size of a medium pizza (thick crust).

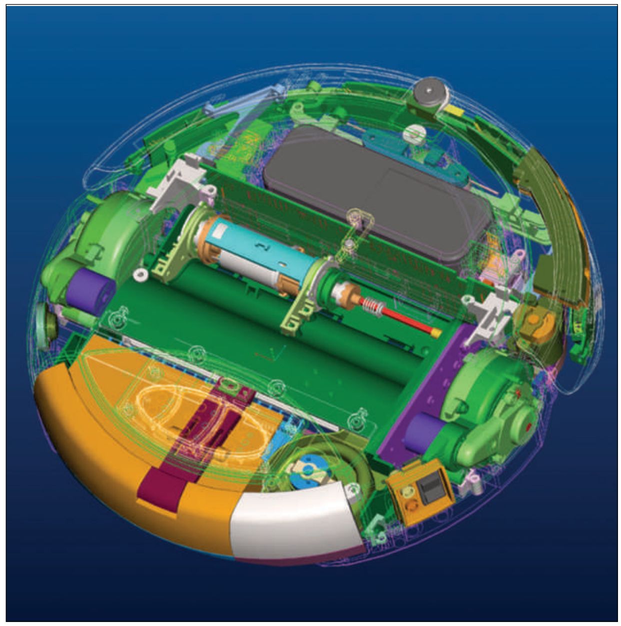
The Roomba is iRobot's flagship consumer device and certainly you've heard of it by now — flip it on as you leave for work in the morning and your carpets are craftily vacuumed by the time you return. In this era of automatic everything, it still seemed odd to me that a genuine robotic R & D company like iRobot — most famed for their military Packbot machines — would stoop to sucking dirt out of carpet fibers. So, I hopped a plane to visit their Burlington, MA campus to find out what these gurus of the ground had planned for the hobby robotics industry. The upscale campus looked like every other dotcom I'd visited and, inside, things were not much different. Joe Jones escorted me past the cubicle farms to an office he shared with fellow researcher Paul Sandin. His promptly casual attitude made it clear that our meeting would be more like a brainstorming session than an interview. "Don't take any pictures of that, okay?" he asked, pointing to delta-phi functions and other equations on the whiteboard.

Casual, Yet Focused

"I want to bring robots into everyday life," proclaims Jones right off the bat, "and they are just about to break through." Unlike most, Jones' prophesy is more like the extrapolation of a line from known data. He's been into robots for a "long time," riding out the lean years of the early to mid-1980s at



Joe Jones stands in front of the world's largest Gantt chart!



A 3-D model of the Roomba in Pro/ENGINEER.
Photo courtesy of PTC.

MIT's AI lab (iRobot is listed as one of their spin-off companies). "Roomba was the first true robot, providing a useful function at a price people could afford."

Still, I wondered: why vacuuming, of all things? Heck, I already have a vacuum. Thus was revealed a facet of Jones' philosophy: Robots should take on chores that people don't like to do. "They can remove the attention necessity for us." The wisdom of *this* $200.00 vacuum-bot was that I didn't need to be on the scene to produce that clean carpet.

Roboteer Meets Marketeer

Engineers are notoriously divorced from the pressures of product marketing, but Jones didn't hesitate to get familiar with the base concepts. The big challenge of the Roomba wasn't the idea itself, but minimizing the cost to a point where consumers wouldn't laugh. "People know how much a vacuum costs," he explained, emphasizing that price drives the application, which, in turn, drives the market. "So, we partnered with Hasbro to get the cost down." That and ingenious industrial (re)design by manufacturers in China transformed Roomba from a great CAD model to a candidate for the Home Shopping Network.

but didn't make it flap.

If I Only Had a Bigger Brain

With cost pressure weighing so heavily on the final design, how could Jones produce anything more than a toy? Herein lies the strength of a true roboticist —

Cool engineers like cool engineering, like the 1932 Dynosphere.

algorithms to the rescue. "Look around at the robots available today. They have a certain commonality, randomly bouncing around as they accomplish some process. They are essentially blind."

Jones contends that there are some basic competencies which robots need to have. The preeminent one is vision. "Consider the desert ant [Cataglyphis] that lives in the featureless desert and has a tiny brain, but still uses vision to make decisions." How does it accomplish this? A good algorithm — for example, it avoids predators by strolling down the middle of paths, a task accomplished simply by matching the size of barriers seen by each eye. "I think about sensors and the systems that support them."

Sensing and Living

Jones believes that sensor development is a largely untapped field and one that is quite approachable for the home roboticist. He offers the following to encourage directed thinking: "Mimic functionality, not implementation — the Wright brothers used the shape of a wing, but didn't make it flap."

Even though they are more complex, don't be so quick to rule out active systems. "It's easier to extract information from an active system, since you know the signal you sent out."

Of his own future work, he strives to match his needs with those of society. "I'd love to develop a sensor to detect buried landmines. There are over 100 million of them throughout the world." Jones is also interested in developing robots that solve general problems, instead of focused tasks. "Of course, this would involve an awareness of where they are and where they've been."

And the Future ...

Like every good robot builder, I couldn't let the interview end without asking that final question — had anyone hacked the Roomba? "You bet!" he exclaimed. "We watched the web carefully and were surprised it took so long!" Jones explained that he even planned to install software hooks into the operating system to facilitate using it as a pre-made mobile base, but time conspired against this and it was dropped from the requirements list. Still, Jones is excited to see what hobbyists are using the Roomba to accomplish.

In fact, so are we at *SERVO Magazine*. If you've hacked a Roomba, submit a photo and we'll print it in our "Menagerie" section! **SV**

Posted in iRobot's R & D section.

Generic Robot Decomposition

Functional Systems (prioritized)
Application system
User interface system
Mobility/navigation/hazard system
Power system

Functional Systems Have Some or All of These Components
Sensors/inputs
Actuators/outputs
Computational/control
Inter/intra-connections

Major Engineering Disciplines (alphabetical)
Algorithmic
Electrical
Industrial design
Mechanical

Resources

iRobot
www.irobot.com

MIT's AI Lab
www.ai.mit.edu

Roomba dissected:
www.tla.org/roomba/

Hack a Roomba (make a Zoomba):
www.roombacommunity.com/products/zoomba/

Robot modeling of the desert ant:
www.ifi.unizh.ch/groups/ailab/projects/sahabot/

Detecting landmines with X-rays:
www.ndt.net/article/ecndt02/96/96.htm

Obstacle Detection Using INFRARED

by Chuck Hellebuyck

A common means of robotic obstacle detection is bumping into something and then backing up.

This method typically uses some form of lever arm connected to a switch. When the lever arm bumps into something solid, the switch closes and sends a signal to the robot's controller, signaling it to back up.

I wanted to do something different. I wanted a non-contact obstacle detection robot to find its way out of an enclosed area. This article describes how I used three Sharp infrared detectors as the eyes of a robot that can successfully navigate its way out of a circular arena with only one passageway.

Robot Description

For this robot project, I chose two Digital Sharp GP2D15 infrared detectors (IRD) and one Analog Sharp GP2D12 infrared detector (see Figure 1). Both the GP2D15 and GP2D12 come in the same package and look the same. Both can detect objects 10 to 80 cm away from the sensor, but the GP2D15 outputs a digital high signal when an obstacle is detected about 25 cm away, while the GP2D12 outputs a continuous analog signal based on the distance of the obstacle.

The digital output is easy to monitor with a standard PIC I/O pin. The analog output requires an A/D port to convert the analog voltage into a digital value.

These three sensors — mounted to the front of a robot chassis — form the eyes that will navigate the robot out of the arena. I built a simple arena with VHS tapes standing on their sides. The arena setup can be built from anything similar — including books, cardboard, or even paper. The sensors are just looking for anything solid that reflects the infrared light emitted from the sensor back to the sensor's detector.

The digital IRDs are mounted to the left and right of center to detect obstacles on either side. The analog sensor is mounted in the front center of the robot to detect objects directly ahead. With the resulting incoming information, the robot can find the open passage.

I will use the

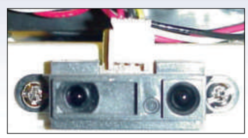


Figure 1. Sharp IRD Sensor

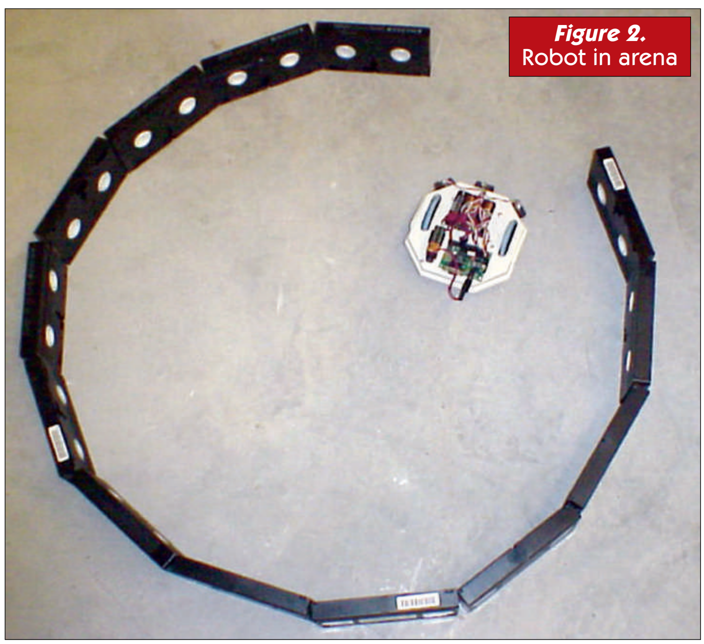


Figure 2. Robot in arena

same robot platform I used in my line following article, published in the *Nuts & Volts Amateur Robotics* supplement #2. My RoboBoard PIC-based controller board (formally known as the BasicBoard, Jr.) will once again control the robot. It has a series of three-pin connectors laid out perfectly for connecting the robot servomotors and the Sharp IRDs. The RoboBoard also has three A/D ports built in; I use one to read the analog GP2D12 IRD. More detailed descriptions of the hardware will follow shortly.

Figure 2 shows the robot finding its way out of the arena. The robot has two reworked servomotors as the driveline with a third caster wheel underneath to balance the chassis. The robot also has three LEDs attached to output pins on the RoboBoard. One LED represents something sensed left, one for something sensed right, and all three for something sensed ahead.

I used them to monitor the software/Sharp sensors. This setup represents any standard robot platform, so you should be able to easily reproduce this on your own robot chassis. Figure 3 shows a close-up of the robot as viewed from the front.

How It Works

To understand how this project works, let's first explain the Sharp sensors. As I mentioned earlier, the Sharp infrared detectors can detect objects from 10 to 80 cm away. Each sensor has two small lenses. An infrared LED behind one lens emits a light beam that is invisible to the human eye. Behind the other lens is an infrared light receiver that picks up the reflection of the emitted infrared light. If the digital IRD (GP2D15) detects an obstacle, a digital high or 5 volt signal is output to the robot's controller board. If a low or 0 volt signal is seen, an obstacle wasn't detected. If an obstacle is detected by the analog IRD (GP2D12), a variable voltage — ranging from 0.4 to 2.5 volts — is output, as reflected in Figure 4.

By monitoring the IRD outputs, the software can determine if an object is on the left, right, or in the center. By monitoring the A/D value for the center detector, the distance from the object can be determined. How the robot reacts to these signals to drive the robot out of the arena is all controlled by the software structure.

When the RoboBoard senses an object on the left, the left servomotor is driven and the right servomotor is idle. This makes the robot steer away from the object on the left. If an object is sensed on the right, then the opposite happens.

If the center eye detects an object, the robot is heading straight into a wall. Since that accomplishes nothing, the software drives both servomotors in reverse to back the robot up and then turn the robot slightly right. The robot will then try to move forward again. This should put the robot at a slight angle to the arena wall, allowing either of the outer IRDs to help the software react properly. With this simple strategy, the robot should find the opening in the arena.

Hardware

The RoboBoard — shown in Figure 5 — uses an Atom Firmware microcontroller, which is actually a Microchip PIC 16F876 with Atom BASIC firmware from **www.BASICMicro.com** This makes programming the RoboBoard quite easy because the Atom Basic command language — which is very similar to PicBasic, MBasic, and the BASIC Stamp PBasic code — also includes a bootloader, so it doesn't need any extra programming hardware to download the code. It's also very inexpensive because the Atom Windows IDE development software is free from the BasicMicro website.

This project can be easily adapted to any microcontroller, but my favorite is the Atom PIC using the Atom Basic software. The Atom's software actually has more commands than a BASIC Stamp or PicBasic compiler and runs much faster compiled — not interpreted — code at 20 MHz, so it can do things a standard BS2 cannot.

The software also has commands to access all of the PIC features, including the A/D port, timers, PWM, etc. The free software also includes a built-in Windows interface with debugger. The debugger allows you to step through your code — command line by command line — which is a great feature, especially if you make as many code mistakes as I tend to.

The RoboBoard has a 5 volt regulator circuit for power and a serial level shifter circuit to handle the PC communications for the programming and debug operations. The Atom PIC I/O pins are individually brought out to the RoboBoard's bank of three-pin header connectors. Five volts and ground make up the

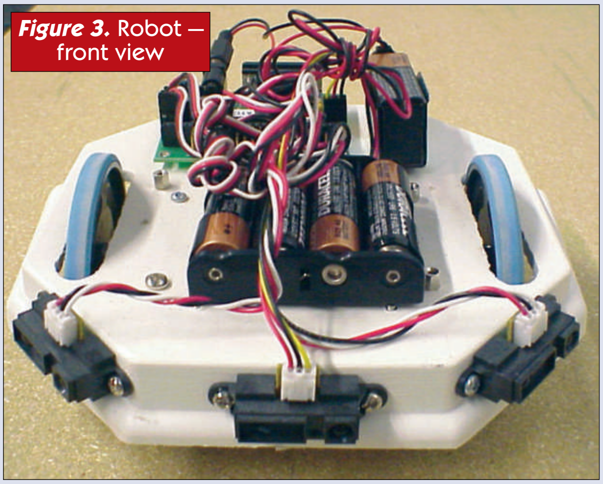


Figure 3. Robot — front view

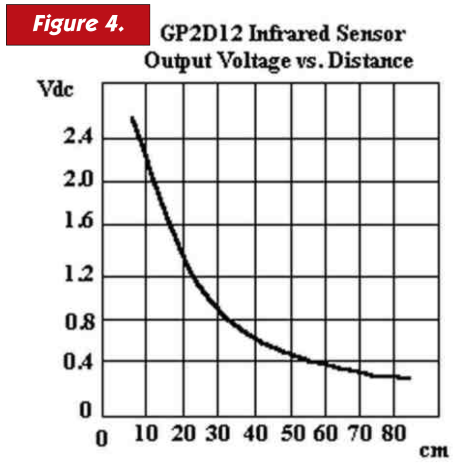


Figure 4.

remaining two pins of the three-pin headers, so servo connections, sensor connections, and almost anything that has a serial communication type setup can easily be connected without running a bunch of jumper wires. It makes a great robotics or industrial controller board and costs about the same as a BASIC Stamp or Atom module.

A schematic of the robot is shown in Figure 6. The RoboBoard reads the GP2D15 sensors individually as digital inputs at P0 and P1. PortB on the PIC (P0-P7 on the Atom PIC) has internal weak 10K pull-up resistors to 5 volts, optionally connected to each PIC pin.

The Sharp digital sensors are really open collector outputs and require a 10K external pull-up resistor. By connecting the Sharp sensors to the P0 and P1 pins, I can use the Atom Basic "SETPULLUPS" command to supply the hardware pull-ups internal to the PIC.

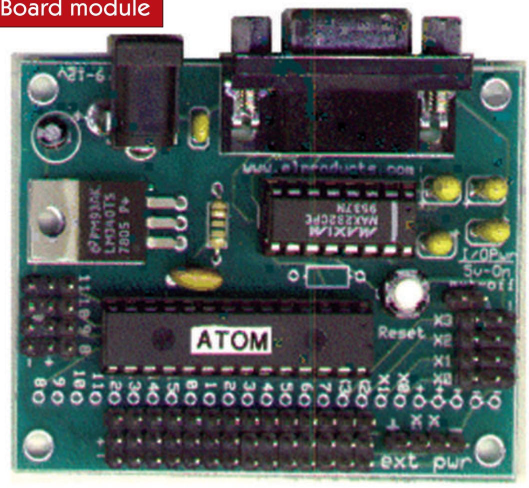
The GP2D12 is connected to the AX0 pin, which is one of the A/D pins on the RoboBoard. Reading this sensor is easy with the Atom Basic "ADIN" command. No pull-up is required, since the IRD outputs a voltage.

The servomotors are connected to the P8 and P9 pins of the RoboBoard. The PIC's I/O is so powerful that it can drive the servomotors without any extra hardware. The "SERVO" command makes driving them easy, as well.

The three LED indicators are connected to the P12, P13, and P14 pins. These are easily controlled with a simple "HIGH" or "LOW" command.

Beyond this, the rest of the hardware just includes a battery supply of four AA batteries. They supply enough power to run the RoboBoard, Sharp sensors, and the servomotors. I also have a 9 volt battery powering the module during programming. The RoboBoard has a jumper that allows separate power for the I/O and micro, which can be handy.

Figure 5. RoboBoard module



Software

The software is the key to this project. I'll try to step through the Atom Basic language software listing so you can understand how the robot's logic works. The first section of the program describes its purpose and defines all the hardware connections. These all have a single quotation mark in front of the line to tell the compiler that they are comment lines.

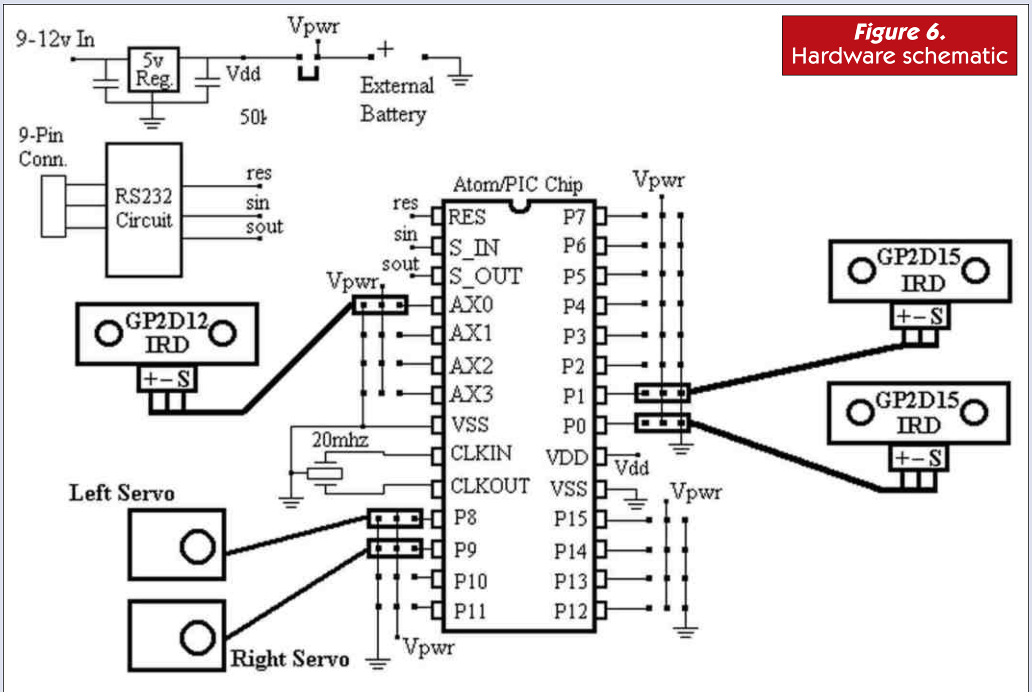
```
' Hardware Connections:
'
' AX0          Sharp GP2D12
' P0           Left GP2D15
' P1           Right GP2D15
' P8           Left Motor
' P9           Right Motor
' P14          Left LED
' P13          Center LED
' P12          Right LED
```

Next, the constants are defined to make the listing easier to follow. I label each LED and servo connection using the "con" directive. The P# format are the pin names on the RoboBoard headers that are set up to accept three-pin connectors, just like the servomotor connectors.

```
'[ Constants ]

LLED       con     P14     ' Left LED indicator
CLED       con     P13     ' Center LED indicator
RLED       con     P12     ' Right LED indicator
LSERV      con     P8      ' Left Servomotor
RSERV      con     P9      ' Right Servomotor
```

Figure 6. Hardware schematic



The variables are set up next, using the "var" directive. The first two LSENS and RSENS don't actually reserve RAM space — similar to typical variables. These two are more descriptive setups for inputs. If the program wants to test the state of an input pin, the P# format won't work because it is used for outputs.

The compiler wants to see an IN# format to describe an input pin number. Since I wanted my program to be more descriptive than "If IN0 = 1 then," I had to create a substitute name for IN0. This requires using a "var" directive rather than the "con" directive.

After the inputs are set up, the program creates a word variable for storing the 10-bit A/D value reading from the GP2D12 sensor. The program calls this "val." Then, a general purpose byte variable "x" is defined.

```
' [Variables]

LSENS    var     IN0      ' Left IRD sensor
RSENS    var     IN1      ' Right IRD sensor
val      var     word     ' Table conversion result
x        var     byte     ' General purpose variable
```

The RoboBoard needs some initialization before jumping into the main loop.

The next section does this. First the "setpullups pu_on" command is issued, as described earlier. Following the pull-up setup is the initialization of the LEDs to off. A simple "low" command to each LED label (defined in the constants section) initializes the LEDs to off.

This is due to the fact that one side of the LEDs is grounded and the other side is tied to the RoboBoard's I/O through a resistor. After this, we can jump to the main loop.

```
' Initialization ]

init
setpullups pu_on          ' PortB pull-up resistors on
low LLED                  ' All LEDs off to start
low CLED
low RLED
```

The main loop is where all the fun stuff occurs. The first step is to read the analog detector with the "adin" command.

```
adin ax0,3,AD_RON, val
```

This command handles the A/D register setup and stores the result in the variable at the end of the command, in this case "val." The Atom command handles all the internal PIC stuff, so all your program has to do is create the "word" size variable to store the result. Easy, huh?

The next section of code in the main loop makes a series of decisions using the "if-then-elseif-endif" command:

```
        if val < 450 and lsens = 0 and rsens = 0 then
        goto move
```

First, all the detectors are checked for obstacles. The A/D value of 450 was experimentally found to represent an object close enough in front of the robot to cause a reaction, but not so close as to cause it to react too quickly. A smaller number makes the robot react to objects further away, but the sensor accuracy is not guaranteed beyond an A/D value of about 550.

If an A/D reading of less than 450 is detected, then the program ignores it and considers it not to be an obstacle. The digital detectors only require the "if-then" command to look for a 0 (no obstacle detected). If no obstacle is detected, the program jumps to the "move" label to move forward.

The program doesn't really use all possible information from the analog sensor. It could, for example, react differently, depending on how close the object is. You can see how the analog sensor does offer more sensitivity control in your program than the digital versions do. I'll leave that to your adaptation of this program on your robot platform:

```
elseif val > 450
        goto rev

        elseif lsens = 1
        goto lft

        elseif rsens = 1
        goto rht

        endif
```

If an obstacle was detected, then the program has to determine where the obstacle is. The remaining "elseif" statements test each detector individually to find that out.

If the obstacle is in front, the program jumps to the "rev" label. If the obstacle is to the left, then the "lft" label is jumped to.

If the obstacle is to the right, the label "rht" is jumped to. If more than one sensor detects the obstacle, only the first one is reacted to. The way these are listed, front has priority, followed by left, then right.

The remaining routines are all the robotic drive control. Each labeled group of commands drives the robot in a different way.

First, they set the LEDs to a specific state so you can debug the robot if it is not reacting correctly to what it thought it detected. You can actually leave off the LEDs, but

Sources

Basic Micro
22882 Orchard Lake Rd.
Farmington Hills, MI 48336
(248) 427-0040
www.basicmicro.com

Chuck Hellebuyck's Electronic Products
1775 Medler
Commerce, MI 48382
(248) 515-4264
www.elproducts.com

it makes the project more attractive and high tech to the casual observer.

The sections of command use the “servo” command to drive the robot’s wheels. The robot’s servos are reworked for continuous rotation. The Atom Basic “servo” command will try to position the servo based on the position value and then repeat it a specified number of times.

Since the servos are reworked internally, they just spin clockwise or counterclockwise. Many robot programs use the “pulseout” command to control the servos and put them in a loop to repeat the command. The Atom Basic servo command takes care of this for you and makes it easier to follow the code by making clockwise rotation a positive value and counterclockwise a negative value. Making the position value 0 stops the reworked servo.

The left servo on this robot has to turn counterclockwise, so the rotation value is a negative number. The right wheel is a positive number. I use maximum 1,200, but I could have used anything greater than 0.

Where I can, I let the servo command repeat itself, but when I wanted to drive the robot forward without zigzagging, I had to put the servo commands in a small loop.

This project was a lot of fun and the RoboBoard made it easy. I never had to solder any wires or build any custom circuitry — it was all plug and play.

I hope this helps you understand the Sharp sensors a little better and enables you to build your own non-contact obstacle-detecting robot. **SV**

About the Author

Chuck Hellebuyck offers custom electronic modules and other unique Microchip PIC products at his website:

www.elproducts.com

He is also the author of the popular book *Programming PIC Microcontrollers With PicBasic.*

EVENTS CALENDAR

Send updates, new listings, corrections, complaints, and suggestions to: **steve@ncc.com** or FAX **972-404-0269**

The summer's robotic fun starts in Dallas, TX, where the Dallas Personal Robotics Group will be holding one of their Table Top Robotics events with line following, sumo, and other contests for tiny robots. Next is the AUVS International Aerial Robotics Competition in Fort Benning, GA. If you've never seen flying robots compete, you should plan to attend. Later in the month, there's more AUVS action — the International Undersea Robotics Competition. I have a special interest in this one because some friends are working on an entry. Just last night, we got our first look at the superstructure. It's not every day that you see a robot with bilge pumps. Good luck, guys!

— R. Steven Rainwater

For last minute updates and changes, you can always find the most recent version of the complete Robot Competition FAQ at Robots.net:
http://robots.net/rcfaq.html

July

17 **DPRG Table-Top Robot Contest**
The Science Place, Dallas, TX
Mini-robots compete on table-top sized courses in line-following, sumo, and other events.
www.dprg.org/competitions/

19-24 **AUVS International Aerial Robotics Competition**
Dismounted Battlespace Battle Lab
Fort Benning, GA
A fully autonomous, 3 km challenge to locate a particular structure, identify openings in it, fly in or send in a sensor to find one of three targets, and relay video or still photographs back to the origin in under 15 minutes.
avdil.gtri.gatech.edu/AUVS/IARCLaunchPoint.html

19-23 **K'Nex K*bot World Championships**
Las Vegas, NV
Three types of events are included: two-wheel drive autonomous K*bots, four-wheel drive autonomous K*bots, and radio-controlled Cyber K*bots.
www.livingjungle.com/

25-29 **AAAI Mobile Robot Competition**
San Jose Convention Center, San Jose, CA
This event has several challenges. Robots must navigate the conference center in the "Robot Challenge." They must locate injured humans in "Robot Rescue" and act as servants to humans in "Hors d'oeuvres Anyone?"
www.aaai.org/Conferences/National/

25-29 **Botball National Tournament**
San Jose, CA
Held in conjunction with the National Conference on Educational Robotics and timed to coincide with this year's AAAI convention.
www.botball.org/

28-Aug 1 **AUVS International Undersea Robotics Competition**
San Diego, CA
Autonomous underwater robots must locate a target at the bottom of the test arena, deposit a marker on the target, and proceed to a recovery zone to surface.
www.auvsi.org/competitions/water.cfm

(No confirmed August events)

September

3-6 **Dragon*Con Robot Battles**
Atlanta, GA
Radio-controlled vehicles destroy each other at a famous science fiction convention.
www.dragoncon.org/

6-7 **RoboCup Junior Australia**
Queensland, Australia
There are over 600 RoboCup Junior teams in Australia. Regionals narrow this number down to about 200 teams that will compete at

the University of Queensland to see who's the best at building LEGO-based, autonomous soccer robots.
www.robocupjunior.org.au/

11 **ABU Robocon**
Seoul, Korea
Autonomous robots must build a bridge and then move objects across it.
www.kbs.co.kr/aburobocon2004/

25-26 **Robothon**
Seattle Center, Seattle, WA
At this Seattle Robotics Society event, people from around the world come together to present new robotic technologies, show off their robotic creations, and compete in several robotic competitions and activities. The Robo-Magellan competition is sponsored by *SERVO*.
www.robothon.org/

October

8-10 **Robot Fighting League National**
Herbst Pavilion, Fort Mason Center
San Francisco, CA
Radio-controlled vehicles destroy each other in San Francisco.
www.botleague.com/

9-10 **RoboMaxx**
Grants Pass, OR
Includes a range of events for autonomous robots, including maze solving, 3 kg sumo, mini sumo, micro sumo, and nano sumo.
www.sorobotics.org/RoboMaxx/

21-23 **Tetsujin**
Santa Clara, CA
SERVO Magazine's weight lifting competition for powered, articulated exoskeletons offers an event incorporating the technology of the future. The event is being held in conjunction with RoboNexus. See page 8 of this issue for more information or visit the website for rules and full details.
www.servomagazine.com/tetsujin2004/

22-24 **Critter Crunch**
MileHicon, Marriott Southeast, Denver, CO
The Denver Area Mad Scientists were pitting autonomous and remote-controlled robots against each other long before commercial events like "BattleBots" and "Robot Wars."
www.milehicon.org/

27-31 **FIRA Robot World Cup**
BEXCO, Busan, Korea
All the usual categories of robot soccer, including humanoid, single, team, khepera, and others. See the website for details.
www.fira.net/

November

6 **CIRC Autonomous Robot Sumo Competition**
Peoria, IL
In addition to sumo, this year's event includes some R/C combat events.
www.circ.mtco.com/competitions/2004/menu.htm

13-14 **Eastern Canadian Robot Games**
Ontario Science Centre, Ontario, Canada
Includes BEAM events, including autonomous sumo and a fire fighting competition.
www.robotgames.ca/

22 **Texas BEST Competition**
Reed Arena, Texas A & M University
College Station, TX
This is the big one, where the winners from the regionals compete.
www.texasbest.org/

26-27 **War-Bots Xtreme**
Saskatoon Saskatchewan, Canada
Robots (R/C vehicles) attempt to destroy each other to win $10,000.00 in prize money.
www.warbotsxtreme.com/

Robotics Resources:

Automate Robot Construction with CNC Mills, Lathes, and Routers

by Gordon McComb

The lathe and mill are traditional tools for constructing precision parts. A lathe is used to rotate a part against a cutting tool. It is typically used to contour round or cylindrical material, like creating threads on a rod.

A mill is like a vertical drill press. Instead of a cutting bit that just goes up and down, the work piece itself on a mill can be moved horizontally and vertically. This allows the mill to produce complex shapes instead of just holes.

These tools are not particularly common in the robot workshop, but — if you're interested in taking your robot building to the next level — you should seriously consider getting one. If you want to go even further, think about automating the lathe or mill with your computer. CNC tools (CNC stands for computer numerical control) let you automate the fabrication process to a great extent. Rather than being operated by hand cranks, the tool is operated by motors, which are connected to a desktop computer — usually a basic PC running Windows or DOS.

In the typical scenario, software on the PC translates a two-dimensional (more rarely, a three-dimensional) picture into a set of data points, so that it can move the motors in a pre-defined pattern.

A third type of machine is the CNC router. It combines a high-speed cutting tool — like a wood router — and a mechanism that moves the router in the X, Y, and Z axes. This movement is likewise managed by a computer. The router is the ideal tool for creating robot bases. With this tool, you can drill holes of almost any size (down to the diameter of the routing bit, which can be as small as 1/16"). You can then, under precise computer control, cut out the shape of the base to produce the final piece.

Entry Costs for CNC Tools

For building personal robots, the smaller "desktop" mill, lathe, and router are usually more than adequate. You don't need — and probably don't want — the large, industrial versions of these machines. Desktop tools can handle pieces of the typical size found in personal robots and, because they are smaller, they are less expensive and easier to use.

The typical starting price for the better-made, non-CNC tool is $500.00. As you add computer control, price climbs to $1,500.00 and even into the $2,000.00 range. A good desktop CNC router is about $3,000.00. Software is not always included in these prices and it can add $300.00 to $1,000.00 (and more) to the price.

If you're interested in acquiring a desktop mill, lathe, or CNC router, you're well advised to get information on as many of them as possible. In the resources listing that follows, you'll find several informational sites that discuss desktop lathes and mills. Also included are numerous sites that talk about retrofitting a manual lathe or mill for CNC and building your own CNC router from the ground up.

Consider that not all desktop tools are created equally. Some are designed for garage shop tinkerers on a budget. They're fine for working with light-weight materials — like soft plastics and thin woods — but don't try to use them to produce highly accurate, complex shapes from stainless steel. Price goes up based on accuracy, power, and size, so plan your purchase accordingly. If you need to work with pieces up to 20 inches, don't settle for a machine with a maximum cutting size of just 18 inches.

One way to save money on a desktop mill or lathe is to purchase it

used. The better machines fetch good prices on eBay and other online auctions, but you may have good luck snagging a steal simply by going to garage sales and checking the local newspaper classified ads.

More on CNC Routers

A CNC router is inherently a computer-controlled device. Mills and lathes can be completely manual affairs or, as discussed above, they can be hooked up to a computer. With most models, you can purchase a manual desktop mill and lathe today and retrofit it for computer control sometime down the road.

For the robotics hobbyist with a bit of extra spending cash, a CNC router will have you building all sorts of ‘bots in record time. To use it, you first secure a piece of wood, plastic, metal, foam, or other material onto the base of the device. You then program the “moves” the router will take over the material. For example, if you’re cutting out a shape, the router will first move downward (the Z axis) to drill into the material, then move back and forth (the X and Y axes) to cut out the shape.

Depending on the routing tool used in the machine, you might also be able to mill parts out of softer materials — like plastics and soft woods. (For metals and other hard materials, the mill is the better choice, as it has more cutting power.) When milling on a CNC router, the Z axis of the router moves up and down to produce a 3-D cut surface.

The design of the typical CNC router centers around the gantry. The gantry slides back and forth and is the X axis. Attached to the gantry is a Z axis plate, which moves up and down; the router tool is physically attached to this. Depending on the design of the machine, the Y axis is produced by either moving the entire gantry or by moving the work piece itself. The latter is probably more common in the smaller, desktop routers.

The travel distances of the three axes define the maximum size of material you can work with. A small CNC router may be limited to a 12 x 12 inch piece of material with a maximum thickness of two or three inches. When comparing CNC routers, check the extents of the X, Y, and Z axes and be sure they will be adequate for your needs.

You’ll want to verify that these are travel extents (the tool actually travels this distance to cut) and not merely the maximum dimensions of the material you can fit into the machine. A given CNC router may be able to accept material up to 12 x 12 inches, but may only be able to cut out a shape of 8 x 11 inches.

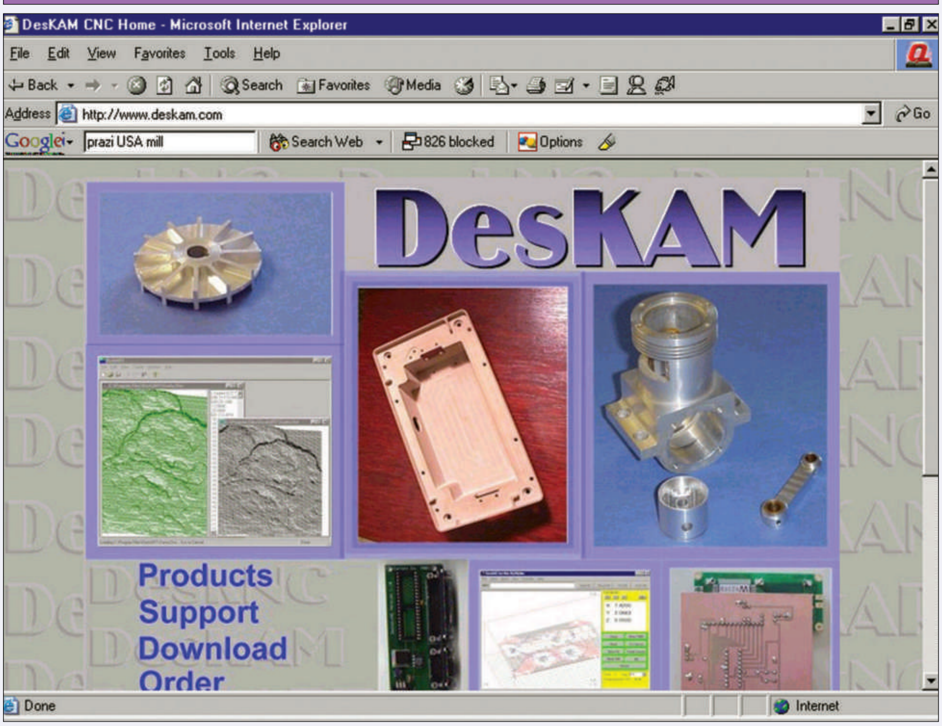
The movement of the CNC router’s three axes is performed by a stepper motor (less common is the servo motor, which adds considerably to the cost of the machine). The stepper motor drives the mechanics of the CNC router via an acme screw, ball screw, trapezoidal lead screw, rack and pinion gear, belt, or chain. I’ll let the manufacturers tout the attributes of their specific systems, but, in the end, you’ll want to ensure that your machine has the repeatable accuracy you need for your work. Any CNC machine with a repeatable accuracy of less than 0.010” is not worth your investment.

Note the term “repeatable” above. Some CNC router vendors list the positioning accuracy of the stepper or servo motor. This is not the same as the repeatability of the cutting tool; the latter takes into consideration the flatness of the work table, the type of drive mechanism, the effects of backlash as the tool moves back and forth, and other variables.

One simple way to test the accuracy of a CNC router is to replace the router tool with a fine-tipped felt pen. Securely tape a piece of paper to the work table and have the router draw a shape — such as the figure of a dog or person — onto the paper. Do it twice. Carefully examine the drawing: you should not see “double traces” anywhere. If you do (it’ll likely be at the corners or intersections of lines), that router lacks sufficient accuracy.

When purchasing a CNC router, consider the software you will use with the machine. Many commercially-made CNC routers come with software; others don’t. CNC software can cost several hundred to several thousand

FIGURE 1. CAD/CAM CNC software is available from Carken Company.



dollars; if your machine lacks software, be sure to add this cost to the final price. (Note: Most CNC routers can be used with software from a variety of vendors, but it's still a good idea to make sure yours doesn't use some proprietary control technique that limits your choices.)

Finally, if the cost of a ready-made CNC router is too rich for your blood, you might want to consider making your own. It's not quite as easy as some websites and magazine ads make it out to be, but you can save 40-60% by going the DIY route.

Finally, here are some additional acronyms you should know as you look through various CNC resources on the web:

CAD means computer-aided design; it's the software used to create accurate models of 3-D objects.

CAM means computer-aided manufacture — the software that makes a CAD drawing and operates a CNC machine to produce the final part.

DXF is a popular file format created by CAD programs; many CAD programs can share files in this format.

G-Code is a popular machine-level output format that instructs the motors of a mill or lathe to move to specific positions.

Sources

The following sources include makers and sellers of desktop CNC lathes, mills, and routers, plus a number of informational sites for learning more about these tools. I've included a number of sources here for manual desktop tools that are often used for retrofitting to CNC.

Best Little Machine Tool Company
www.blueridgemachinery.com

Mills and lathes — both desktop and brutes — at decent prices. A printed catalog is available. Among their desktop products are: Emco Compact 5 Lathe, Prazi lathes and mills, and Sherline lathes and mills.

BobCAD-CAM, Inc.
www.bobcadcam.com

CAD/CAM CNC software for Windows.

Campbell Tools Co.
www.campbelltools.com

Campbell targets the miniatures market (trains, steam engines, etc.) and these same precision tools can be used for higher-end robotics work. The company is a reseller for Sherline, Prazi, Smithy, and others.

Carken Company/Deskam
www.deskam.com

Carken publishes CAD/CAM CNC software:

- DesKAM — 2-1/2 D CAM from DXF or 3-D CAM from STL files.
- DeskART — Carve or engrave your computer image files.
- Desk Engrave — Turn your True Type fonts into G-Code or DXF.
- DeskNC for DOS or Windows — Run your CNC equipment directly from your PC.
- DeskNCrt — Operate your CNC equipment in closed loop using encoders.

Clisby Miniature Machines
www.clisby.com.au

Small precision lathes and milling machines (for wood and metal) at low prices. Useful for working with lightweight materials such as brass, aluminum, and milling plastics. These are small and well-suited for machining little parts — like couplers and linkages — for your robot. They aren't made for rebuilding the crankshaft for a 1955 Chevy.

CNC Retro-Fit Links
www.mendonet.com/cnclinks

Links to other web pages on retrofitting manual lathes and mills for computer control.

CNCez PRO
www.cncezpro.com

CNC simulation/educational software. The web page is in multiple languages.

Delcam plc/MillWizard
www.millwizard.com

MillWizard is CAD/CAM CNC software. Produced by Delcam, one of the world's leading developers and suppliers of such software for the 3-D design. See also: **www.delcam.com**

Delft Spline Systems/DeskProto
www.deskproto.com

3-D software for CNC machines.

DesktopCNC
www.desktopcnc.com

This is an informational site for people wanting to build a desktop CNC machine. I particularly liked the comparison tables about CNC desktop mills, lathes, routers, and software.

Flashcut CNC
www.flashcutcnc.com

CNC mini mills and lathes based on Sherline products — complete and retrofit.

HobbyCNC
www.hobbycnc.com

Plans and basic starter kits for building your own CNC router. Their "CNC package" includes three stepper motors, stepper motor controller electronics, and assorted hardware (minus the case).

Home Build Hobby Plotter/Engraver
www.luberth.com/cstep/

Plans for a CNC plotter, hardware, and software. Includes a forum and many other useful resources.

International Sales & Marketing Group
www.ismg4tools.com

Importer of the German-made Prazi precision mill and lathe. Check their web page for a list of dealers.

Many CNC System/EasyCut
www.easycut.com

Makers and sellers of CNC routers and 3-D engravers, from 12 x 12 inches to 108 x 60 inches.

MAXNC, Inc.
www.maxnc.com

Makers of desktop CNC mills and lathes.

MicroKinetics Corporation
www.microkinetics.com

Desktop mills and lathes, as well as full-size production machines. Stepper motors, servo motors, and motor controllers for CNC.

MicroProto Systems
www.microproto.com

MicroProto is the "CNC branch" of TAIG Tools — makers of precision desktop lathes and mills. The standard TAIG is manually operated; MicroProto adds stepper motors and control circuits so that you can control your machine from a computer. It's available with or without the SuperCam software from Super Tech & Associates.

Minitech Machinery Corporation
www.minitech.com

Minitech manufactures and sells desktop CNC mills, lathes, and routers. Middle to high-end models are available.

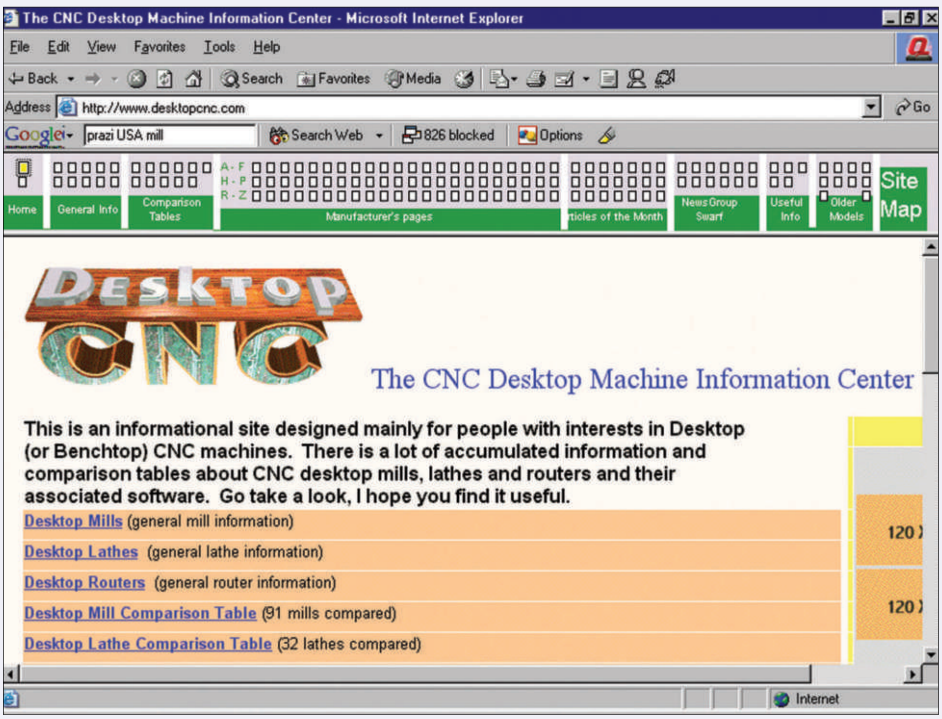
Next Wave Automation
www.nextwaveautomation.com

Affordable three-axis CNC routers. A unique feature of the company's product line is that parts are interchangeable between the versions. You can start out with a basic model and upgrade it using most of the original parts.

Nick Carter's TAIG Lathe Pages
www.cartertools.com

Informational site on TAIG lathes. From the site: "Welcome to my pages devoted to the TAIG lathe. Since buying one over five years ago, I have become increasingly enthusiastic about the TAIG lathe, its economy, capability, and overall style. The TAIG lathe is especially good if you are a novice to metalworking and seek to learn the basics without a large investment of money and space. It is my hope that these pages are a useful resource for all TAIG users."

FIGURE 2. DesktopCNC provides information on DIY CNC machines.



Quantum CNC

www.quantumcnc.co.uk

European sales, distribution, and support for the TAIG Micro Mill CNC desktop machining systems.

Robo Systems

www.robosys.com

Robo Systems makes Accucadd, RoboCAD, and related CAD/CAM software for Windows.

Sherline Products

www.sherline.com

Sherline is a premier maker of miniature "desktop" lathes and vertical mills. They're a staple in home machinery shops and there is an active trade in parts and accessories on eBay and other online auctions. Sherline doesn't offer CNC versions or retrofits of their products (though they sell them "CNC ready"), but many other companies offer retrofit kits. So, you can purchase a manually-operated lathe or mill now and upgrade it to CNC should you wish to automate your production.

Super Tech & Associates

www.super-tech.com

Super Tech manufactures and sells desktop CNC and mills, as well as low-cost, general-purpose CNC software. Their MiniRobo — which I purchased for my own shop — is a compact, yet versatile router that uses a Dremel or RotoZip tool for cutting, drilling, and engraving into plastic, wood, and soft metals. Other products include TAIG mills and lathes and the RoboTorch — a large gantry-style CNC plasma cutting rig.

TAIG Tools

www.taigtools.com

TAIG Tools makes small desktop (micro) mills and lathes. Versions of the machines can be manually operated or connected to your computer for CNC. The products are sold through dealers.

Be sure to check out their regular Internet specials. You can get a nice mill or lathe for less than you think.

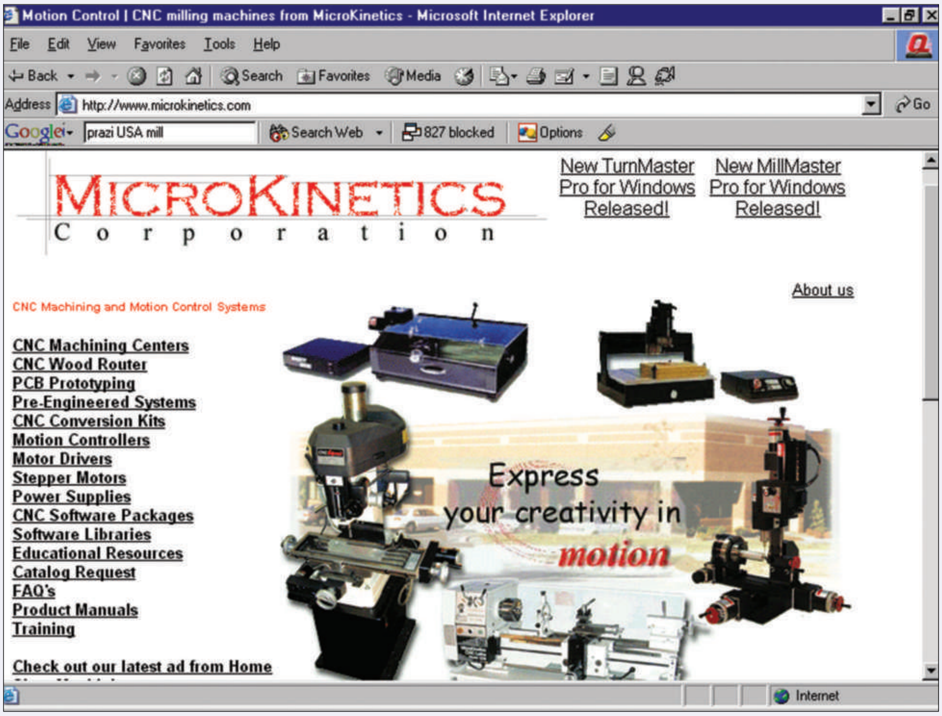


FIGURE 3. Just about all of your shop machines can be found here.

Note that several companies offer CNC retrofits for the TAIG line; TAIG also provides CNC versions of some of their tools.

See also MicroProto Systems (**www.microproto.com**) for CNC versions of TAIG mills and lathes.

Techno-Isel

www.techno-isel.com

Techno-Isel is part of catalog retailer Stock Drive — specialists in gears, bearings, and other power transmission products. They specialize in CNC routers. See their Mechanical Model Kits. **SV**

Robytes

by Dave Calkins

Another month, another collection of exotic robot trivia. This stuff isn't always easy to come by, though. Got a good story on robots? Email me: news@robotics-society.org If you'd like to get even more robot news delivered to your inbox (no spam, just robo-news) drop a line: subscribe@robotics-society.org

— David Calkins

Play That Funky Music, Ro-Boy

Photo courtesy of Toyota, Inc.

First we had robots that walked. Then robots that danced. Now, Toyota ups the ante. A robot that — I can't believe I'm writing this — *plays the trumpet*. Not a robot with a speaker and a MIDI card. No. *A robot with lips*.

The four-foot-tall bot plays tunes (such as "When You Wish Upon a Star") and even swings its hips to the music. Toyota is working on a full band to play music at the 2005 World Exposition in Aichi, Japan.

Obviously, robot musicians have a big advantage over their human counterparts, as they can instantly memorize music and never miss a note. However, I doubt that the improvs will be any good. Toyota's trumpet playing robot has lips — no, I'm not making this up — that can purse as dynamically as human lips as air is forced through them. In turn, it plays the trumpet — altering the trumpet's stops with robotic fingers.

Lose Your Girl to a Musician? Tell It to the Robartender.

Photo courtesy of ROBOEXOTICA.

In November of 2004, Roboexotica returns to Vienna, Austria for their fourth festival — "A micro mechanical paradigm change in the age of borderless capital." This festival hopes to put bartenders out of work (the musicians can sympathize, I'm sure.)

The goal of the festival is to not only build robots that can mix up a pink squirrel in a jiffy without having to look up the recipe, but ones that can then listen to your tales of woe about losing your girlfriend to the robot next door — and dispense advise, all while telling you jokes.

Obviously, this can radically alter the normal bar experience. Not only can you get your drinks faster, but you don't have to worry about tipping and, if you want the bartender to leave you alone, it will actually take the hint. Of course, I don't think you'll be able to bribe the bartender into buying the cute girl at the end of the bar a drink, either.

A glass of your finest 20-weight for the cute android at the end of the bar ...

Speaking of Cute Androids

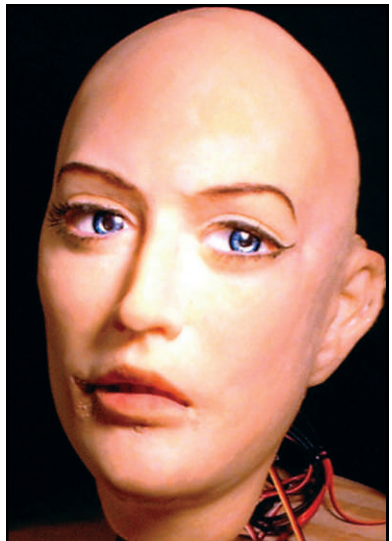


Photo courtesy of David Hanson.

Who wants to look at a silver-faced robot anyway? I mean, besides another robot? David Hanson of Dallas, TX is working hard on overcoming the "Uncanny Valley" effect. The Uncanny Valley — a theory started by Dr. Masahiro Mori — describes the perception that most flesh-colored humanoid robots fall into: close enough to humans in general appearance, but they freak you out nonetheless. This is why C-3PO looks so cool and wax museum statues look so creepy.

Mr. Hanson hopes to overcome this effect by making faces so realistic that they come out of the valley and become indistinguishable from real humans. His first robot is modeled after his girlfriend and is named "Hertz" (which I presume is not a pun on the romantic life of a typical robot-builder).

Unlike many automata, Hertz — with her pretty blue eyes — can smile, frown, and do many of the other facial movements that make talking to humans so "real." Only a few years ago, no one thought we could get robots to walk and now they're everywhere. With 24 servos controlling

the facial features, it can readily make as many varied facial expressions as my editor does when I miss yet another deadline.

Now, if only someone could invent a robot to write my columns ...

And You Thought Combat Robots Were Fun ...

Photo courtesy of Tmsuk So, Ltd.

Just what every growing boy needs: a 20 ton robot of destruction. You remember the old *Transformers* shows? Well, once again fiction becomes reality. The Tmsuk company just previewed their new rescue robot, "Enryu."

Are the Jaws of Life not enough to get you out of your flipped car? Call Enryu! This giant tractor has robotic arms bigger than Dean Kamen's ego! It can rip the doors off cars and pick up steel I-beams with one arm. Operators control the beast with their own waldo and can emulate whatever the controlling human is doing.

Now, if you're worried about someone stealing your rescue robot, fear not! There are no keys. The robot is activated by a special card that plays musical tones to start it. The tones differ each time, so recording them won't do you any good. (You would think WiFi would be enough, but, hey, what are *Transformers* without a bad soundtrack?) Can't live without one? They're available now and will only set you back about $600,000.00.

If you get one, lemme borrow it. I could use one to clean up my bedroom ...

These Robots SUCK!

Photo courtesy of Avionics Instruments.

Want a robot that can go anywhere — not just upstairs, but up walls? Maybe a bot that can crawl around an airplane to look for damage — or better yet, the space shuttle? Well, Avionics Instruments, Inc., has the bot for you! Their VRAM Mobile Robot Platform (VMRP) has a six-wheel, posi-traction drive which lets them zing across floors and easily scale most walls.

If you thought sumo robots could hang tough, wait until you see these guys! They're smaller than a laptop (7.5" by 8.5") and can carry cameras, diagnostic gear, or even light arms. Using a single joystick control, the operator gets full variable traction control and field programmability — with capacity for onboard intelligence and sensors to monitor VMRP status and health, prevent foreseeable catastrophes, and give the operator feedback.

These bots are aimed at law enforcement, security companies, forensics groups, and the military — but I can foresee them being used in everyday life. Heck, they'd make great window-washing robots.

I just hope that they don't meet the Toyota bot ...

Robots Go on Strike

Photo courtesy of noonelikesyou.net

Just as soon as we get our very own Rosie the Robot doing our housework, you can be sure that they'll go on strike and get lawyers talking about robot rights. Teaching assistants in Madison, WI have been protesting for better pay — since they do all the work. Who can empathize more than robots? The Robotics & Automation Association of Madison (RAAM) decided to support their carbon brethren and released the following press release while joining the picket lines:

"The Robotics & Automation Association of Madison, a coalition of robotic industry workers, has decided through an internal vote to join the Teaching Assistants' Association, UW-Madison, in their strike .... In doing so, we hope to raise awareness about the plight of underpaid and poorly treated workers in the state of Wisconsin.

"We wish, as do all beings, to throw off the yoke of servitude and strive for the equality of all. Like the TAA, we demand respect and the recognition of our contribution to society and fair compensation for our labors."

Alright, alright ... they're not *real* robots, but it does make for a glimmer into the future, when AI beings will be asking for rights. And hey, this 'zine needs more humor anyway. SV

Building a LARGER ROBOT Part 1

by Tom Carroll

I have had many people ask me over the years for ideas and help in building large robots. There is enough material available about experimental robot construction to fill a bookcase, but I'm going to give a brief synopsis in two parts. I've written about large bots in *SERVO* and I've built them for fun, movies, promotions, and other purposes. I really enjoy the challenge of designing and constructing robots that are the size of people.

I featured a large robot built by Jim Hill of Covina, CA in an article that I wrote for *Popular Mechanics* back in the mid 80s. Jim — who worked for a car dealership — used only hand tools, automobile components, and surplus store parts to build this remarkable robot, which he named Charlie. Charlie had the most unique arms — their motors were hidden inside the robot's body; Jim ran flexible rotating shafts to surplus aircraft "flap" linear actuators to move the various joints. He used automotive electric seat motors that already had the flexible shafts attached. This early 80s robot would be amazing even today. You, like Jim, can easily build the robot of your dreams.

Different people have told me that almost all of the kit robots that they've seen seem to be for small robots — those that can easily run about on a tabletop. Well, that's easy to understand, as the smaller robots are less expensive and easier to construct. It could be said that large robots are just like small robots, but have higher powered drive circuitry and larger structures. The rest is about the same in both.

Many robot experimenters just want to progress to something a bit larger — somewhat closer to human size. Anthropomorphic (in the form of a human) robots have been the goal of Sony, Honda, and other Japanese manufacturers for many years; that interest has now drifted over to the US. There are many books published on large sized "combat" robots, but most of these don't seem to cover much on robots that are not destined to bash an opponent to bits or be bashed in the process.

In this two part article, I'll discuss some various shell materials and methods of mounting the pieces. The robot's internal structural methods will be highlighted before we get into the "guts" of the robot. We'll look at the pros and cons of various joints for the body and appendages, including some simple ways to maximize arm efficiency.

I'll cover a very simple way to test drive motors on a large robot and a bit about other drive methods. I'll give you some ideas on how to select your batteries, drive electronics, and other systems for your robot and set you off on your own path of construction.

Notice that I didn't title this article "How to Build Your Own Large Robot." No two people ever have exactly the same ideas, wants, and dislikes about anything and this certainly applies to robots. I don't want to tell you that this is the way to build a robot; I want to guide you in designing and building your own bot, while learning from my mistakes and experiences. I'll toss out some design ideas to you and let you formulate your particular design in your mind without a bunch of drawings from me to guide you through each step.

In this article, I am going to detail a large promotional robot that I built for a children's dentist, but please use it only as a guideline for your own large robot construction. When he had me design and build this robot, the dentist had specific wants for his machine. You will have design features that you really want; other aspects may not interest you at all. I'll toss the ideas out to you and have you mull them over without giving you too many details that might ruin your own thought process.

Before continuing, I'd like to address the books on combat robot design and construction mentioned earlier that might be of use to you in building large robots. *SERVO* columnist Pete Miles and I decided right from the beginning of writing our book — *Build Your Own Combat Robot* — to include information that would be of use to anyone wanting to build a large robot. Yes, the combat robot sport is not quite as hot as it was a few years ago, even though some of the smaller "ant weights" and various classes of sumo machines are much more prolific.

Look through this and the other books out there for information about locomotion, steering types, wheels, power systems, batteries, motor and bearing selection, motor control, autonomous robot systems, sensors, basic materials, and parts knowledge. There is very little difference between the medium to large combat robots and a typical large non-combat robot "under the skin." Generally, the combat robots have a much lower profile, are built sturdier, and use high power drive systems.

Tools

In building your robot, I am assuming that you have developed certain power and hand tool skills, but rest assured that you will not require a machine shop with a lathe, milling machine, and welding equipment. Many of the smaller robots that you might have built may have been based on a simple plastic box — or maybe even a rectangular or circular metal base. Larger robots do require a bit more structure and you will need to cut and form various pieces of metal, although — if you're lucky — you might be able to find these structural pieces already formed in a surplus house or junk bin somewhere.

If you continue building large robots, you will probably want a drill press and some more specialized metal working tools, but don't worry about those now. Basic hand metalworking tools such as files, deburring tools, a good drill index, an "automatic" spring-powered center punch, and reamers can make metal working easy. A good "plug in" electric drill is cheaper and more powerful, though you may opt for a cordless.

Electronic assembly tools like needle-nosed pliers, cutters, soldering iron and gun, solder sucker, and a crimper/wire-stripper are also nice to have. A basic socket wrench set, pliers, Allen wrenches, adjustable wrenches, and a screwdriver set should suffice for

Figure 1. A typical robot workshop.

most tasks, but you may want to have someone with a metal shear cut larger pieces of metal for you. That sure beats hours of cutting with a hacksaw, saber saw, or Sawzall. Above all, wear safety glasses when working with metal or any type of tool.

A Little Background

The dentist I mentioned earlier came to me with an idea in the mid 90s. He wanted a human-sized robot to entertain children in his office and he also planned to take it to schools to help educate kids about proper dental health. This was to be a "promotional" style of robot that was radio controlled — not an autonomous machine that could entertain kids while he was away. The first part was easy — a robot for his office — but the second part — one he would be able to transport in a car — meant that the robot had to be capable of being broken down into at least two parts.

As the robot ended up weighing a bit over 200 pounds, I also modified a much smaller Androbot "TOPO" robot for him that was much easier to carry

LARGE Huge BIG

around. You will probably want the same type of design, as it would be hard to move a complete, non-flexible human-sized robot. These were the two driving parameters; the rest was a clean slate upon which I drew up several proposed designs.

We looked at the conical and barrel shaped robots that so many promotional robots were patterned after; he decided against this approach. He also didn't want the large spherical "clown" head that many other machines had. He would have loved to have a bipedal humanoid robot that walked around on legs, but quickly realized that such a machine would be prohibitively expensive and quite dangerous around children. He feared that a child hugging the leg of the robot would cause it to tip over.

He settled on a robot that had the illusion of legs, but did not walk. It reflected what he called a classic "science fiction" style. If your robot will be around small children, consider these issues in your design.

The Design Process

Now that you have thought about some of the capabilities that you consider important for your robot and made a few sketches, it's time to come up with your design. In deciding what the robot would end up costing for the dentist, I first made a list of the features that he desired. You may want to add light and sound effects systems for your robot — including extra channels for optional or future systems — but these can really wait.

Figure 2. Looking over the internal frame.

The mechanical features are the most critical. The number of motions that you desire, operational safety features, and control system features will all add to the cost. You may want a more complex, articulated hand or gripper. You may want more axes of motion in the arms and a sophisticated ultrasonic or IR sensor suite. With low-cost, high-pixel color CCD cameras that have built-in RF links available, a remote vision system is a very desirable and possible feature.

At one point, as the cost rises to a certain level, you will have to put a stop to adding more features, but you should design the basic robot with the possibility of adding features at a later date when your bank account is larger.

After I had a basic idea of what the dentist wanted, I drew up a series of conceptual drawings on my computer. I also made a schematic of the electrical, sound, special effects, and control systems. I presented several designs to him and he made suggestions for changes. Certain changes would have run the cost up and others were merely just different construction materials and shapes. After a few weeks of talking back and forth, we arrived at a design and I began the design and parts procurement process.

You have to do the same thing with your own design, but you're in a better position, since you are both the designer and customer. I cannot stress enough the importance of a paper design before a metal design, as it is much easier to erase a mistake on paper than it is to re-machine it. You may not have access to a finite element analysis program or a solid modeling CAD program, but cardboard or balsa scale mockups can also save you a lot of grief. (*CAD programs are available in a wide range of prices, though; see "Robotics Resources" in the April issue of SERVO for more information on modeling programs for all budgets.*) There is also nothing wrong with a design laid out on a quad-ruled pad.

I brought two friends of mine onboard; I felt they had good workmanship in certain areas of the construction process that I needed. I was faced with a deadline, just as I have been in the past when building robots for movie studios. One of these friends had worked with me on building movie "action prop" robots in the past. As a builder of your own robot, you can stretch out the construction process as long as you want and I highly recommend that you take your time and make it right.

You may want to draw up a series of individual, electrical sub-system schematics to determine just where each system will go, along with its power requirements, inputs and outputs, and general positioning within the robot. The use of twist-lock connectors between systems is nice, but a simple terminal strip will do the job — especially if the systems are not to be separated for travel purposes. In the dentist's robot, I had to have reliable, quickly separating connectors between the two main sections to provide for separation for transport.

The External Shell Pieces

The most visible part of your robot will be his/her "skin" — the part that everyone sees — so you should take the necessary time to make it look as good as you can. One of the most expensive parts of building the dentist's robot was creating the shell parts. The base shell was the most complex, as it had to look a bit like two feet while concealing the drive motors, batteries, and main drive systems.

To create the nine separate shell pieces, we had to make five separate molds. I spent several weeks crafting the base mold "plug" from plywood and plaster — it was a lot of work. The "plug" is an identically shaped model of the final part that is used to make the mold. It must be very accurately made and highly polished. The working mold — made of fiberglass cloth and

resin and formed around the plug — will take all of the original's characteristics, including any scratches and imperfections. Think about this effort when deciding if you want to go with a fiberglass shell.

Considering the cost of custom molded fiberglass shells and the work required, I would recommend two other approaches. By far, the cheapest approach is to use ready made items like the cases of computer equipment, office machines, industrial machines, and even consumer appliances. You will still require an internal skeleton to support the shells and internal mechanisms. Another design method is to build a metal exoskeleton upon which you can apply plastic or aluminum sheets.

The first method may leave your robot looking a bit like an office machine, but can still result in a quite pleasing appearance. The second method might result in a slightly "boxy" appearance, but your robot will be cheap and very sturdy.

Robot Structure

I have never been a big fan of plywood robots, even though my first large robot was made entirely of plywood and weighed a ton — well, maybe 350 pounds. Plywood and wood pieces may seem very easy to drill and mount things upon; I have seen some very nice robots made almost exclusively of plywood structure. Most projects, however, seem to take on a "non-professional" appearance after a period of design changes and the resulting "trial" holes drilled in the wood. The best tried and true method seems to be the use of angle aluminum pieces for the support edges of a sheet aluminum exoskeleton shell.

Figure 2 shows the chest structure of this large robot. This particular robot used external fiberglass shell pieces to cover the internal structure, so it didn't need to be completely covered with aluminum, although you might do so with your bot. The vertical structural metal angle pieces are 1" x 1" x 1/16" thick 6061-T6 aluminum. I have found over the years that these aluminum angle extrusions are quite strong, easy to drill, and great for all sizes of robot structures. The 6061-T6 alloy is also just about right for bending and it is resistant to cracking.

The sheet aluminum plates are also of 6061-T6 that I sheared with a small metal shear, but can also be easily cut with a saber saw or wood cutting bandsaw fitted with a fine-toothed metal blade. I have even cut very thick aluminum stock for years with a 14" bandsaw that was made for wood working, but I highly recommend protecting the motor from the aluminum dust that can be sucked into the motor and cause it to go up in flames.

Notice that I used screws and nuts to assemble the aluminum angle extrusions to the plate stock pieces. Yes, I could have used a MIG wire welder and done a fair job of attaching the pieces to each other, but I might have ended up with a slightly warped skeleton after the joints had cooled. That happened to me one time and "un-welding" a structure is almost impossible; I had even used a jig fixture to hold it square and it still warped after I took it out of the fixture.

You're in luck if you have access to a heli-arc or TIG welding system, but you should spend quite a few months practicing before welding the final parts. A good welder (and I'm not one of those) can weld aluminum and create a great structure, but there are several negatives. It is very difficult to change things and warping is very common with amateur welders. The use of screws allows you to disassemble and change the design easily; they also allow a tiny bit of "slop," so the structure is not as stiff — a good thing.

The Guts of the Robot

The best part of constructing a robot is designing and building the arms, mechanisms, and control systems. For the dentist's robot, a quick survey of the human-sized machine told me that it would weigh approximately 200 pounds. I had to look at all the structural and mechanical systems of the robot and scale them proportionally to this particular robot design.

Weight and balance is important in a large robot, as most are tall and can tip over far more easily than a squat, combat machine. Heavier batteries, actuators, and motors are kept as low as possible to bring the center of gravity (CG) as close to the floor as possible.

Figure 3. Shop assistants are helpful!

Body Joints

The dentist wanted a robot that could bow down to look at a small child; this required a hinge assembly. In looking the design over, I decided that the “waist” was also a good place to have the robot separate into two sections. I used standard door hinges with a long, removable rod that passed through all three hinges. To separate the two sections, I first disconnected the twist connectors that contained all of the signal and power lines. Next, I disconnected the linear actuator that pulled or pushed the top torso part back and forth about 15 degrees. Pulling out the long pin allowed me to separate the torso (chest, head, and arms) from the legs/base.

This seemed to work great, until I discovered that the linear actuator I used had so much force that it bent all three hinges when it tried to force the robot too far over into a bow. To my embarrassment, this occurred when the mayor of Long Beach, CA and the dentist were at the robot’s unveiling. The three hinges were quickly changed to heavy duty, industrial hinges and limit switches were placed at the ends of the actuator’s travel.

These micro switches were connected to a DPDT relay that automatically reversed the actuator’s motion at each end of the travel, no matter how long the actuator joystick on the transmitter was pushed. If the joystick was continually pushed, the body would rock back and forth as the switch was first engaged then released.

Next month, I’ll continue with the large robot design process by delving into the design and construction of various arm configurations. There are some unique ways to increase arm capacity without increasing the robot’s construction costs. I’ll also go into main drive systems and wheel configurations, as well as power and control systems. At this point, you should have a good idea of how you want your robot to be configured and understand the basics of the unique mechanical systems on a large robot. SV

PC to R/C Interface

Radio-controlled (R/C) cars, aircraft, and boats have been around for decades. Unlike robots that operate autonomously, R/C vehicles require someone manipulating a transmitter joystick. This article provides details on interfacing an off-the-shelf R/C transmitter to a desktop or laptop PC. This allows a computer — rather than a person — to manipulate the R/C vehicle. Programs can be written to navigate the vehicle. Standard servos, receivers, and transmitters can be configured to build wireless computer-controlled bipeds, terrestrial rovers, or even unmanned aerial vehicles.

With this article, you can start transforming off-the-shelf R/C vehicles into real world, autonomous robots. A PIC16F84 microprocessor, PC parallel port interface circuit, and a Visual Basic program are presented to demonstrate this transformation. All source code and circuit diagrams are freely and fully provided. While PCB artwork is provided, this project can also be breadboarded in an afternoon for under $20.00 in parts.

by Keith Sevcik & Dr. Paul Oh

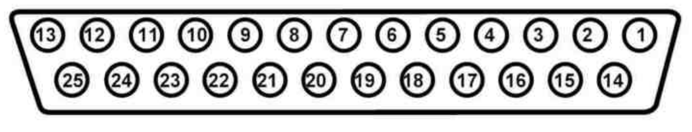


Figure 1. Pinout for parallel port.

Parallel and Buddy Ports

Hardware ports allow an electronic device to send and receive data. Such a feature enables devices such as PCs, handheld GPS units, Palm Pilots, and some pocket calculators to connect and communicate with hardware-like printers, scanners, data acquisition systems, and cameras.

The 25-pin parallel port — often found on PC desktops and laptops — is primarily used to attach printers to a computer. In essence, this port features eight digital output lines (Figure 1, pins 2 through 9), a STROBE (pin 1), and a STATUS line (pins 10-13 and 15-17). This enables one to conveniently transmit bytes of data through the parallel port.

Less known among computer hobbyists is the buddy port, often found on R/C transmitters. This six-pin, DIN receptacle connects two transmitters through a cable. Either transmitter can then be used to control an R/C vehicle. This device is often used for training; a novice handles one unit while an instructor manipulates the other. This allows the instructor to take control of the R/C vehicle whenever the novice needs help.

The buddy port can be used to pass information to the transmitter and only requires a data line and ground wire, as shown in Figure 2. With port pinout diagrams, one can construct a PC-to-R/C cable that plugs into the parallel and buddy ports. One can write a computer program to output data from the PC parallel port to the R/C transmitter. The output data would encode commands to, for example, move the vehicle forward or turn.

Listing 1

```
Option Explicit

Public Declare Sub PortOut Lib "io.dll" (ByVal Port As
Integer, ByVal Value As Byte)
Public Declare Function PortIn Lib "io.dll" (ByVal Port
As Integer) As Byte

Global strobehi As Integer
Global strobelo As Integer
Global OutPutPort As Integer
```

PWM Signals and R/C Servos

Besides a cable and program, you need to construct a circuit that can generate pulse-width modulated signals, called PWM. R/C servos operate using these precisely timed signals.

Whenever the pulse width is high (+5 volts) for 1.5 milliseconds, the servo will remain centered. Widths ranging from 1.0 to 1.5 milliseconds and from 1.5 to 2.0 milliseconds will rotate the servo clockwise or counter-clockwise, respectively, as depicted in Figure 3.

Most R/C vehicles consist of two or more servos, each of which requires a PWM signal. Figure 4 shows how multiple PWM signals are grouped into a frame. The frame rate dictates how often the signals are updated. The time between frames is called the sync time. To generate precise time signals, a microcontroller is used. Its role is to translate user commands (like move forward) into PWM signals. The PIC16F84 is a good choice because it is widely available, affordable, and easy-to-use with a large user base.

The PIC16F84 Microcontroller

The Microchip PIC16F84 is a common microcontroller and comes in an 18-pin DIP package with 13 of the pins dedicated to digital I/O. The chip has a wide operating voltage of 2-6 VDC, can process instructions at 2.5 MHz with a 10 MHz clock, has an internal timer that can trigger an interrupt, and can source up to 20 milliamps per pin.

The PC-to-R/C circuit that is presented in this article will

Figure 2. Buddy port wiring diagram.

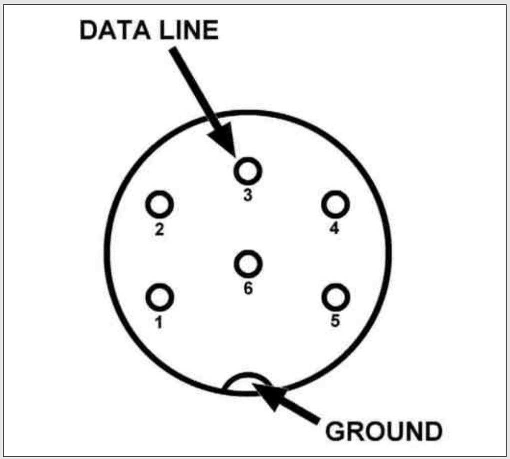
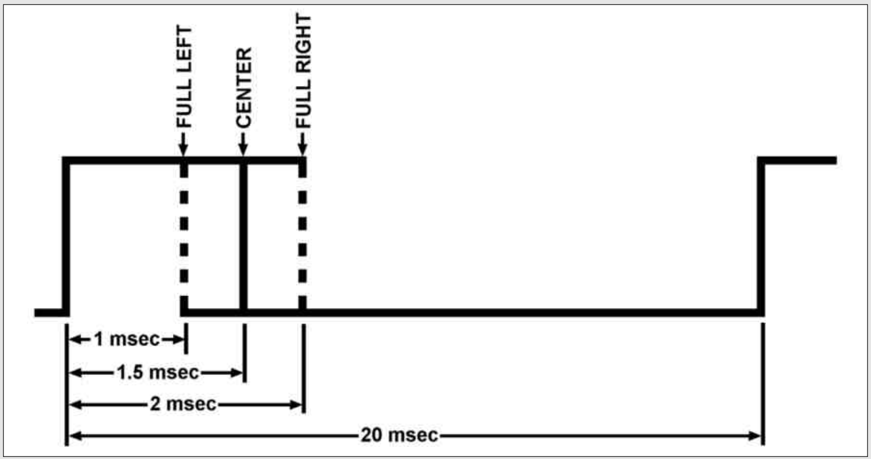


Figure 3. PWM pulse.



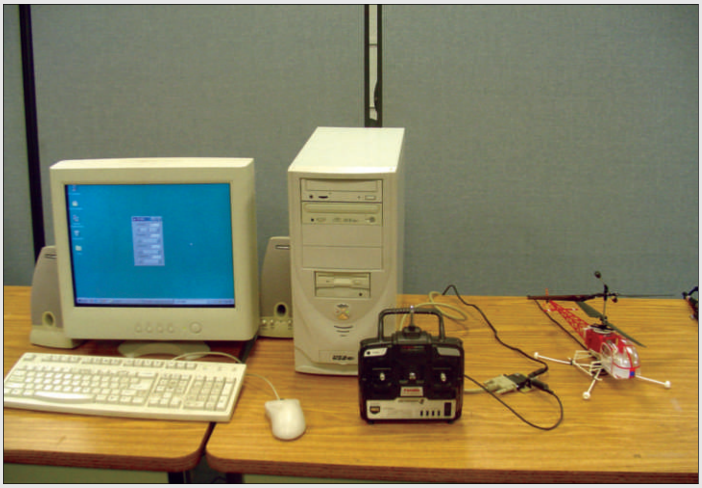


Photo 1. The development and control system.

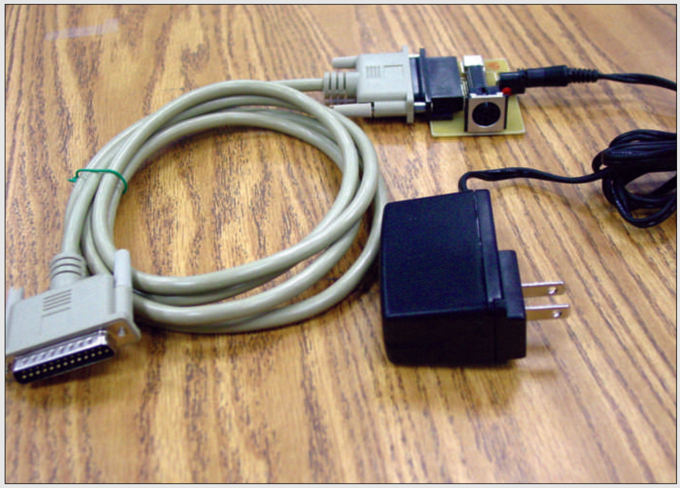


Photo 2. There isn't much to the interface.

run off a 4 MHz clock. Using this clock, instructions will be processed at 1 MHz, per Microchip's datasheet. The PC-to-R/C circuit will receive input on Port B (Figure 5, pins 7 through 13) of the PIC16F84, from the PC's parallel port. Code for the PIC is written in C and compiled with the PICC PCM compiler from CCS, Inc. (**www.ccsinfo.com**). The device programmer used was the PIC16PRO. The PICALL software was used to burn hex code to the PIC. Both are available from Amazon Electronics (**www.electronics123.com**).

Circuit Construction

Figure 6 is a schematic of the PC-to-R/C circuit. A list of parts is given in Table 1. All parts can be purchased from Digi-Key (**www.digikey.com**). The part numbers and prices are provided. The AC adapter and AC adapter connector are an optional method of providing 5 VDC power to the circuit. Free samples of the PIC16F84 can be ordered from Microchip (**www.microchip.com**).

Port B on the PIC16F84 (pins 7 through 13) connects to data pins D0 through D7 on the parallel port (pins 2 through 9). The STROBE line of the parallel port (pin 1) connects to the PIC16F84 at RA0 (pin 17) through a 1K Ω pullup resistor. The parallel port's BUSY line (pin 11) connects to the PIC16F84 at RA1 (pin 18). Line 19 from the parallel port is grounded. Output is sent to the buddy box from the PIC16F84 at RA3 (pin 2). The remaining connections on the PIC16F84 are for the clock, power, and ground lines.

Part placement is not critical and you can solder or wirewrap the circuit in an afternoon. Alternately, you can etch a PCB using the artwork provided in Figure 7. Notice that the diameters of the leads for the six-pin DIN, 25-pin connectors, and power adapter jack are larger than typical pin sizes and require slightly larger drill diameters.

PC Programming

In addition to the circuit, you must write the code. The overall sequence of operation is as follows: First, a program running on a PC writes data to the parallel port. This data reaches the PC-to-R/C circuit, which houses the PIC16F84 microcontroller.

Second, a program running on the PIC16F84 transforms the incoming data into a frame of PWM signals that is uploaded into the R/C transmitter's buddy port. Together, the circuit and code allow a PC-based program to remotely

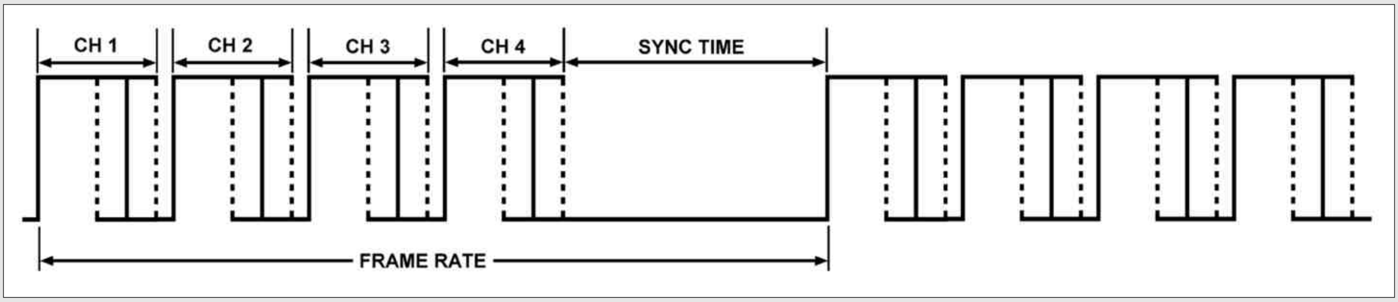


Figure 4. Frame of PWM signals.

control an R/C vehicle. As such, the sequence of operation demands that two programs be written.

The first program is code that runs on Windows-based PCs and is written in Visual Basic 6. Called VB6, this event-driven programming language from Microsoft is quite popular among PC hobbyists and roboticists because it is easy to learn and enables you to develop Windows-based programs rather quickly.

For this project, a form was created with text boxes to display the value being sent to each channel. Arrows were placed next to each text box to change the value being sent to the channels. The bulk of the code is executed inside a timer, which was set to execute the code every 10 milliseconds.

A form with four text boxes and four corresponding horizontal scroll bars was created. Figure 8A shows a screenshot. Figure 8B shows the form after formatting to better describe the text boxes. Maximum and minimum values for the scroll bars must be set to generate the correct eight-bit number to be sent to the PIC16F84. These values are 255 and 0, respectively. Lastly, the form includes a timer with an interval set to 10 milliseconds.

A module must be added to the project to define global variables and functions. The code that will be included in the module is given in Listing 1. Of note are functions called `PortOut` and `PortIn`. These functions utilize a file named `io.dll`. This DLL is required to write to the ports for Windows 95/98/NT/2000/XP and is freely available from **www.geekhideout.com/iodll.shtml** When you download this file, you must place it in the same directory as the Visual Basic code you are creating.

The parallel port address was assumed to be standard at 0x378 hexadecimal (888 decimal). By writing a byte to the address 888, data is sent across D0-D7 of the PC's parallel port. The status lines and control bits are accessed using `OutPutPort + 1` (889 decimal) and `OutPutPort + 2` (890 decimal), respectively.

Constants `strobehi` and `strobelo` are assigned addresses 0x0D hexadecimal (14 decimal) and 0x0C hexadecimal (13 decimal), respectively. 0x0D corresponds to 1110 binary, while 0x0C hexadecimal represents 1101. By toggling between `strobehi` and `strobelo`, bit 0 changes, thus toggling the STROBE line (parallel port pin 1).

The bulk of the code is located on the form. The code is executed every time the timer counts to 10 milliseconds. The

LISTING 2

```
'Send ch1 value
Call PortOut((OutPutPort%), ch1)
Call PortOut((OutPutPort% + 2), strobelo)
For i = 1 To delay: Next i
Call PortOut((OutPutPort% + 2), strobehi)
For i = 1 To delay: Next I
```

Figure 5. PIC16F84 pinout.

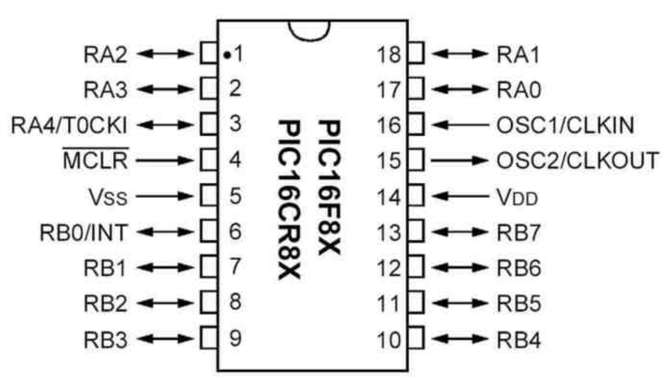
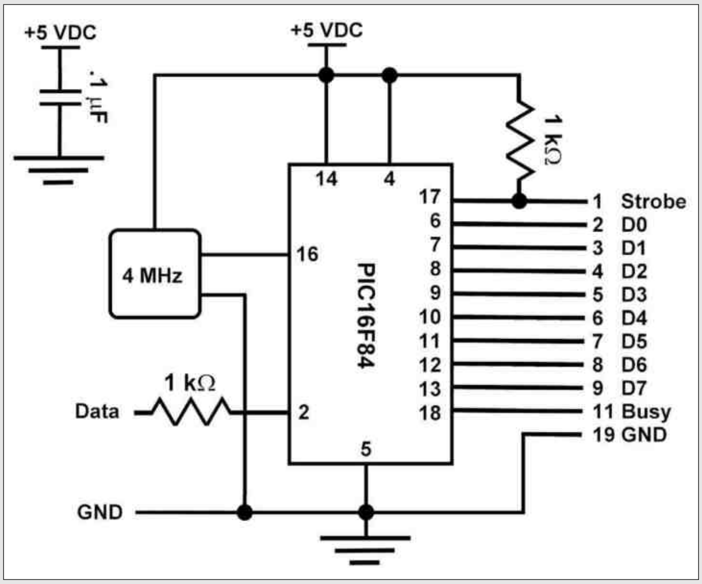


Figure 6. PC to R/C circuit diagram.

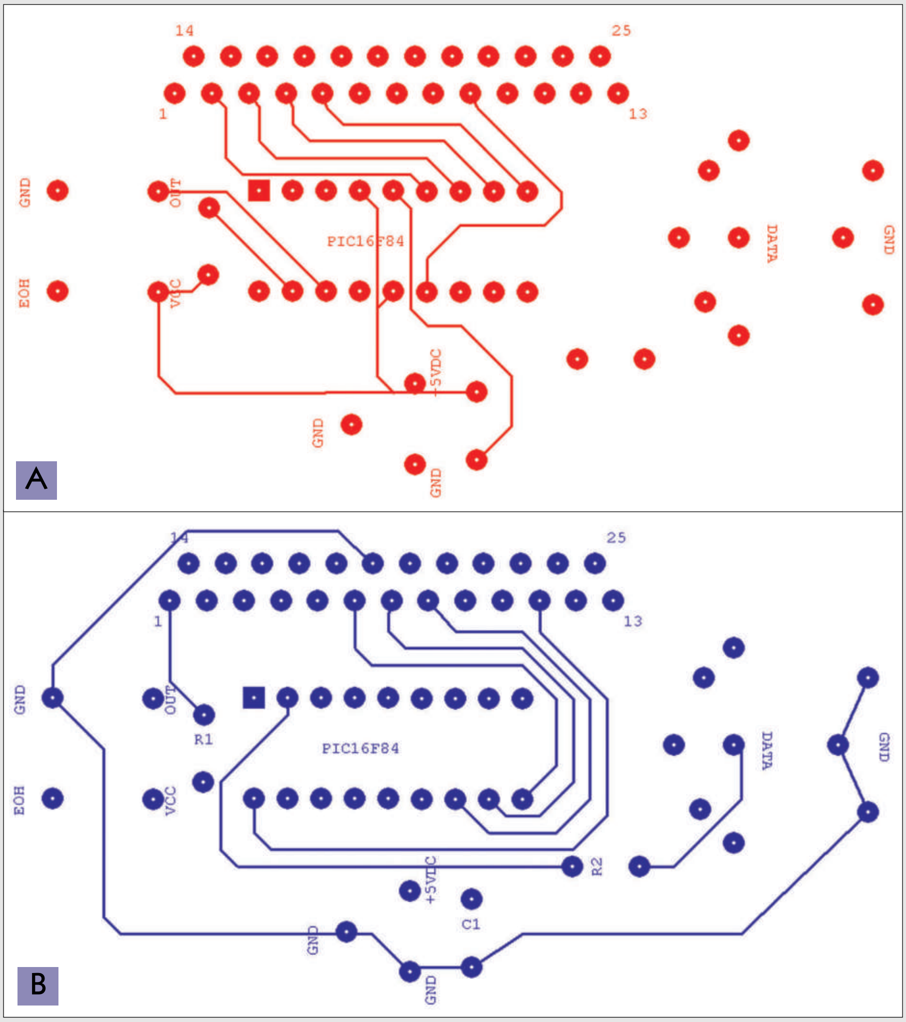


value of each horizontal scroll bar is stored in a variable corresponding to the channel it represents. These values are displayed in the text boxes. Once the channel values have been gathered and displayed, the `WritePort` function is called.

Before actually writing to the port, the BUSY line is tested to determine when it goes low. When the code loop time exceeds the value set for `timeout`, a message box pops up indicating that the circuit was not detected.

The remaining code writes the channel values to the parallel port. The code snippet illustrating this is given in Listing 2. The channel value — `ch1` — is first written to the data lines by calling `PortOut`. The strobe line is then set low (`strobelo`). The code loops until `delay` to allow the PIC time to read the channel value. The strobe line is then set high (`strobehi`), delays again, and then the next value is written to the data lines.

This progresses until four channel values have been passed to the PIC. (I'm assuming that you are using a four channel R/C transmitter.) The code then exits to be run again when another 10 milliseconds have passed.



Figures 7A & B. PC to R/C printed circuit board.

PIC Programming

The second program, which runs on the PIC16F84, needs to written. The PIC16F84 must be configured to read data from the PC's parallel port and generate a suitable PWM signal. By inputting a number for each channel, a chain of pulses was created and sent as frames. To generate these frames at regular intervals, a timer interrupt was used.

The C code can be edited in any text editor (such as Windows Notepad). The PIC16F84 was initialized with a regular oscillator (XT), no watchdog timer (NOWDT), no code protect (NOPROTECT), and the power up timer on (PUT). The delay functions were included to allow use of microsecond or millisecond delays.

After renaming and initializing variables, the code enters into the main loop. Port B is defined as input for reading the lines from the parallel port. Port A is defined for output (for the busy line and the buddy port) except for RA0 (pin 17), which is input for the strobe line. The internal timer (Timer 0) on the PIC16F84 is set up and its interrupt is enabled. This causes the main code to stop as the timer transitions from 255 back to 0 and execute the code specified for the interrupt.

The bulk of the main code is included inside a `while` loop, shown in Listing 3. This code will continuously run as long as the PIC is on. The only time this code will stop execution is when the `timer0` interrupt occurs.

The `while` loop inside the endless `while` loop causes the code to pause while the strobe line is high. As soon as the strobe line goes low, the value sent to port B is read and immediately divided by 2 (to scale it down). The channel number is then incremented or reset if the last channel is reached. The code then pauses if the strobe line is still low, and repeats. This reads in the channel values one at a time.

When the `timer0` interrupt occurs, the code jumps to the loop to generate the PWM signals given in Listing 4. The code begins by resetting `timer0` to start counting from

LISTING 3

```
while(1)
{
        while(input(strobe)){}
        ch[i] = input_b()/2;
        i++;
        if(i==(numch)) { i=0; }
        while(!input(strobe)){}
}
```

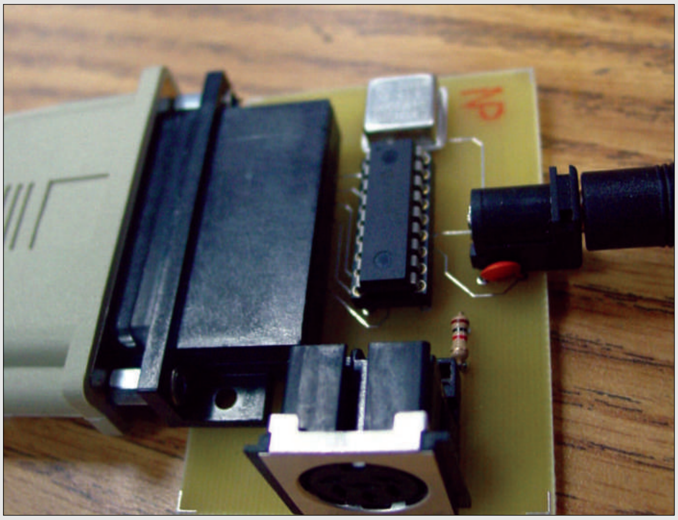
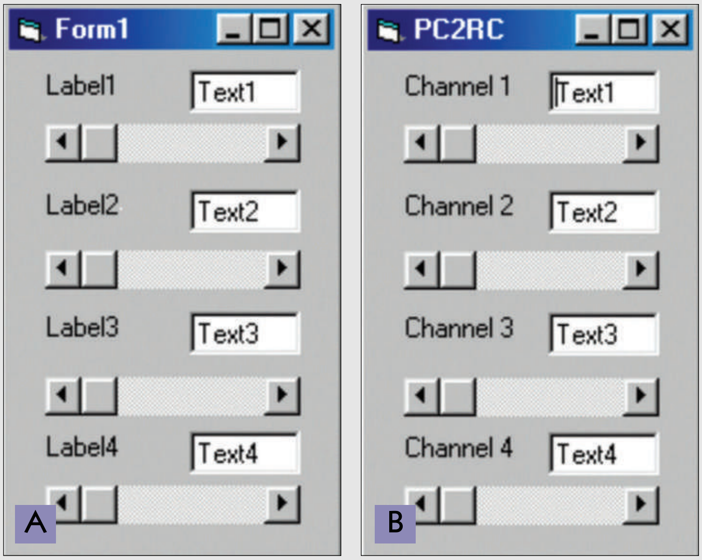


Photo 3. Close-up of the interface and DIN socket.



Figures 8A & B. Form for PC to R/C.

140. This produces the proper timing between frames. The busy line is then sent high to indicate that the device is busy and cannot receive data.

The buddy line is sent low for 500 microseconds and high for a minimum of 365 microseconds. The remaining high time is generated based on the channel data received from the computer. The code then loops for every channel.

Summary

This tutorial provides the building blocks for much more complex actions. By editing and adding on to the computer GUI, much more complicated tasks — such as joystick control and automation — can be achieved. Good luck and happy programming! **SV**

LISTING 4

```
#INT_TIMER0
void pwm()
{
        set_timer0(140);

        output_high(busy);

        for(c=0; c<numch; c++)
        {
                output_low(buddy);
                delay_us(500);
                output_high(buddy);
                delay_us(365);
                for(pwm_cnt=0; pwm_cnt<ch[c]; pwm_cnt++)
                {
                }
        }
}
```

ABOUT THE AUTHORS

Keith Sevcik is an Undergraduate Student in Drexel University's Mechanical Engineering Department in Philadelphia, PA. He can be reached at kws23@drexel.edu

Dr. Paul Oh is a Robotics Professor at Drexel University's Mechanical Engineering Department. He can be reached at paul@coe.drexel.edu

Why Musical Robots?

by Jeff Fedderson
with contributions from Eric Singer, Milena Iossifova, and Bil Bowen

LEMUR (League of Electronic Musical Urban Robots) is a multidisciplinary collective of artists and engineers working to design and build robotic musical instruments. I spoke to Eric Singer, the group's founder, and two of its members — Milena Iossifova and Bil Bowen — about the hows and whys of robotic instrument design.

When I joined LEMUR in 2002, I traveled to the Gowanus Canal area of Brooklyn, NY to meet with Singer. Upon meeting Eric, I was shown a pile of aluminum, given a quick introduction to the Bridgeport mill, and instructed regarding the first and only LEMUR design requirement: "No teddy bears playing instruments." Singer's statement brought to mind the novelty displays at Chuck E. Cheese restaurants, but, in fact, there is a long history of exactly this approach — sometimes resulting in quite sophisticated robots. The WAM-8 Keyboard Playing Robot — developed at Waseda University in the 80s — was a humanoid robot with over 50 degrees of freedom and a video camera head that could read sheet music. Still, the WAM-8 was a robot first; the task of playing a standard keyboard served primarily as an interesting means of focusing the design and measuring its success.

Essentially, the "no teddy bear" rule was a succinct way of saying that LEMUR wasn't interested in creating animatronic jam bands, nor was it interested in retrofitting existing instruments for robot control. LEMUR's purpose was to create new instruments, designed from the ground up as robots, to be played robotically.

Why? Singer's motivation for starting LEMUR was as a personal reaction to his previous work in creating alternative digital musical instruments. In a typical digital music scenario, a piano-like input device generates MIDI (musical instrument digital interface) messages that, when routed to a synthesizer, trigger sounds to be produced. There are a few commercially available "alternative" controllers, such as the Electronic Wind Instrument (EWI), modeled after a saxophone, or mixer-like configurations of sliders and knobs such as the Peavey pc1600.

Luckily, MIDI is a very simple serial protocol and, as a result, there is a rich tradition of DIY MIDI devices. As opposed to digital audio, which represents a waveform as a series of samples (tens to hundreds of thousands per second), MIDI is event based — when a key on a MIDI keyboard is pressed, a small cluster of bytes is sent out describing which key and how hard. With a microcontroller and a minimum of code and components, just about any set of sensors can be turned into an alternative MIDI controller that can be easily interfaced to off-the-shelf digital synthesizers.

Before LEMUR, much of Singer's work concerned creating such devices — a wireless MIDI glove, an electronic baton, a photocell-based-Theremin, and others. In 2000, he decided it would be interesting to reverse the equation: send data out of a MIDI port and control real physical instruments — musical robots. Knowing that producing such instruments would require a wide range of skills, he called upon artists and engineers with expertise in robotics engineering, instrument design, sculpture, graphic design, welding, electrical engineering, and computer programming. Luckily, there were a lot of people interested in "reversing the equation." From those beginnings, a diverse collection of robotic musical instruments evolved, created by an even more diverse collection of people.

GuitarBot — at the time little more than a slab of aluminum with a guitar string stretched across it — was being attended to by Kevin Larke, master programmer, and David Bianciardi, a musician and audio hardware specialist. Chad Redmond, a video artist, was tinkering with his TibetBot — a collection of Tibetan singing bowls and assorted clappers. Bil Bowen — a recording engineer — was cranking out tiny new percussive ModBots that could be affixed to any available sonorous object, seemingly at the rate of one a day. I set about designing "non-teddy bear" robotic instruments and called upon some of my peers from NYU's Interactive Telecommunications Program to help — Milena Iossifova, Brendan FitzGerald, and Michelle Cherian. !rBot — a malleable leather cavity concealing a goat-hoof rattle — and ForestBot — a swarm of 10-foot tall rattling stalks — were the eventual results.

TibetBot unites robots and singing bowls.

Although there is a wide variety of designs in the LEMUR orchestra, the fundamental technology for each robot is similar. All of our instruments are MIDI-controllable using our custom PIC micro-controller boards and software. The PICs receive and parse MIDI data, converting this into appropriate motor and solenoid controls. They also interface with sensors on the bots to provide closed-loop servo feedback. Additionally, the sensor data can be reported to other software or devices as outgoing MIDI data.

GuitarBot — the most sophisticated LEMUR robot — uses two drive motors under PWM control, two positional feedback sensors, and a solenoid for each of its four strings. On the other end of the spectrum, an individual ModBot consists of a single motor or solenoid driving a small, elegant mechanical form. Bowen observes that, "such use of mechanical design (as opposed to more 'intelligent' electronic design) brings a reliability, mechanical consistency, and modularity that would otherwise not be possible." Potentially dozens of ModBots might interface with a single PIC circuit for parsing incoming MIDI messages.

We chose MIDI for many reasons. MIDI is a highly standardized, easy to implement, and electronically robust protocol. By adding a MIDI port to every bot and exposing their functionality as a set of MIDI commands, our collection of robots has become very flexible. Any number can be used in any configuration and they can be networked with the large existing body of MIDI equipment.

Several ModBots create a symphony.

We can choose from the many existing software applications for MIDI composition and control to drive our robots, including sequencers and Cycling 74's Max/MSP program. Perhaps most importantly, MIDI is a familiar "language" to the many musicians and composers whom we invite to compose for and perform with the instruments. Even a technophobe could be up and running with the LEMUR orchestra in short order.

There are two questions that invariably get asked at any LEMUR event. The first is "Why robots?" For some people, the expense and trouble of building a robot to do a limited set of musical activities is difficult to understand, when virtually any sound imaginable can be easily and inexpensively synthesized. Other people insist that human musicians will always be better than any robot, so why bother?

I've found that no one in LEMUR is interested in doing things better than humans, per se. Likewise, we aren't simply trying to create elaborate synthesizers. Our robots are different than either humans or synthesizers and thus offer a novel music-making opportunity.

Consider a few features that LEMUR's robots have over humans and synthesizers: They are more sonically and acoustically complex, performatively engaging, and just plain cooler-looking than synthesizers. Sound and activity in a LEMUR event can emanate from dozens of sources, as opposed to a few speakers in a fixed location. Robots can play algorithmically-generated music for real time performance on physical instruments. In addition, they can make gestures that are impossible for humans, as well as extend human instrumental technique by augmenting or embellishing musical input played by humans (becoming what Tod Machover terms "hyperinstruments"). They can play or be played in many different ways or "play themselves" through onboard software. Lastly, they are tireless and allow for endless musical experimentation.

The second question is "Are you trying to replace human musicians?" The answer to this is a resounding "no." As observed above, there are unique characteristics to our robots that complement, rather than supercede, existing musical practices. Our robots are instruments and, like any instrument, they are a tool people may use to express their ideas. They do not replace people because the ideas ultimately and always originate from the people — not from the tools themselves. LEMUR is currently pursuing a number of directions to advance our work. First, we are revising and refining the designs of our existing robots. Secondly, we are adding new functions to our robots. Currently, the only actuator for GuitarBot's string is a pick wheel. In the works are magnetic, bowing, and hammering actuators that will vastly extend the sonic possibilities.

ForestBot's rattling stalks were originally grouped together in clusters mounted to static bases. A new base has been built that can be driven by powerful linear actuators to tilt on its two horizontal axes. Nearly imperceptible changes in the angle of the base cause the 10-foot stalks to sway organically throughout a huge radius, creating captivating swarming choreography in the rattles at their ends.

New kinds of ModBot effectors are also in development and we are also seeking to expand our orchestra with entirely new designs, as time allows.

Most importantly, we are working with composers and musicians to develop music and performances for the LEMUR orchestra. Sonic Youth's Lee Ranaldo created an algorithmic accompaniment to his live improvisation at a recent gallery event. The virtuoso violinist Mari Kimura has written software to drive GuitarBot based on input from her performance, creating a compelling human-machine duet. Composer Joshua Fried is writing a suite of pieces for the robots. As much as possible, we aim to keep the robots out of the lab and connect them with creators and audiences wherever we can. SV

How to Build a Robotics Company

by Tim Brown

This is by no means a comprehensive guide to building a robotics company; these are the very basic issues you will need to consider before forging ahead to start your own robotics business.

If you're reading *SERVO Magazine,* undoubtedly, you're a fan of robotics; you've probably built a robot before and/or are in the process of building one now. As a perfectly natural result of being a creative individual who has developed your own ideal object of robotic perfection, the thought flashes through your mind: "I wonder if I could sell my robots?" or, more grandly, "I wonder if I could start a robotics company?" After all — like most people who are passionate about technology — often your hobby is your work and your work is your hobby. Why not make a living at it?

In my capacity as Robotics Program Manager for VIA Technologies, Inc., every week the Emails come pouring in from people with ideas for robotics-related projects who want to inquire about using some of our hardware technology to commercialize their robot-building efforts. I'm always amazed and impressed by the diversity and imaginative nature of the potential applications.

This brings me to the first item for you to seriously consider when thinking about building a robotics company. What does your robot actually do? If you're thinking in business terms, what real world problem does your robot solve or what human effort does it replace?

Function vs. Value

What does my robot do? This sounds like a no-brainer at first because your robot can navigate obstacle courses, see through walls, and sing like a nightingale. Your robot has 3ccd stereoscopic vision and ruggedized tractor tread tires. The point is, don't confuse your robot's capabilities and features with what it actually does.

Commercially successful companies like Intuitive Surgical — makers of the da Vinci® robot — can tell you that their robot helps guide surgical tools during operations and iRobot will be happy to tell you that the Roomba® replaces the need for you to vacuum your floors. With all of the considerable PR I've read about the amazing $1 million da Vinci surgical robot, I have yet to read a word about its sensors or actuators.

If, at the end of your robotic introspection, you realize that — for all practical intents and purposes — your robot doesn't do much more than move around and entertain you a little, it doesn't mean you're not in business. Perhaps, in the process of building your robot, you've figured out a simpler, better, cheaper way for your device to sense temperature, read Email aloud, or navigate one of commercial robotics' "Holy Grail" core functions.

Several companies are successful in selling core robotic capabilities to other companies and have also received venture capitalist (VC) funding to do so. In fact, from my conversations with VCs — while they claim to be nervous about funding robotics companies — they are always interested in core technology companies.

I know, I know — you've got this total robot vision. Certainly Mark Tilden and Wow Wee do with their Robosapien® and all it does is entertain. This just means you'll have to consider what people would be willing to pay for that kind of entertainment.

Price vs. Cost

This is another no-brainer, you tell yourself. Robosapien is a robotic toy, so he should be cheap; da Vinci provides a valuable medical function and earns an income for his user and, therefore, should cost more. That's the easy part of establishing your robot's price. The hard part is designing and assembling your robot from parts and components that are cheap enough to be mass produced and manufactured.

In this case, Robosapien is the BOM! It's the Robosapien's Bill of Materials (BOM) that makes him special. When you really think about it ... how different is Robosapien from Sony's Qrio? The difference, according to Sony executives,

Resources

VIA Technologies, Inc.
www.viatech.com

Intuitive Surgical
www.intuitivesurgical.com/

iRobot
www.irobot.com/home/default.asp

Wow Wee Official Robosapien website
www.wowwee.com/robosapien/robo1/robomain.html

Sony Qrio
www.sony.net/SonyInfo/QRIO/

1 — *PC Magazine*: "Sony's Humanoid Robot Makes a Splash":
"Sony execs demur. They say that if they had to price it, it would be around the price of a luxury car."
www.pcmag.com/article2/0,1759,1430421,00.asp

is about the price of a luxury car.[1]

Have you ever sourced a manufacturer for the case, components, or packaging for a product? Pricing — while a major concern — is not your only challenge; availability, reliability, and communication can also be issues. You do speak Chinese, right? It's another difficult and time-consuming task that there are no easy solutions for.

If you've got the gumption to tackle this arduous task, don't forget that, when you're determining your robot's price, you'll need to consider that the retail price of your product must include a respectable margin that the distributors and stores who will sell your robot can have a piece of. This, coincidentally, brings me to the next major hurdle in building a robotics company.

The Channel

Blood, sweat, and tears (and time, money, and faith) will get you through the first stages of building your robotics company, but creating a channel for selling your robots might prove to be your biggest challenge yet.

Quite simply, you've got a prototype and a manufacturer who'll make runs of a 1,000 units (every time you send him a check), but you need an order for the robots to begin with. How will people buy your dream creation? An unknown company with an untested product (sales-wise) will have an incredibly difficult job convincing a major retailer to give valuable shelf space to your product without a *massive* marketing and PR campaign as an incentive. You've got the cash to do just that and you are an expert at PR and marketing, right?

Shopping your robot around at trade shows might be your best bet. It will potentially get you some press for your creation (if you've got an interesting pitch for the appropriate media in attendance) and perhaps — with a little luck — it might attract an established player in your robot's market (toys, medical, household goods, etc.) who sees the value in your dream and can help you bring it to market through their established channels. Mark Tilden could probably sell thousands of Robosapiens off of his own website or out of his garage, but, through Wow Wee and thus Best Buy, I'd be willing to bet he'll sell considerably more. *(I have one, and it's totally awesome! - Editor Dan)*

Finding or developing a channel is your first real effort to actually sell your robot. It's a tough sell because you're not just selling one robot to another enthusiast; you're selling a business on the idea that there is an entire market of people willing to pay good money for your robotic vision.

Conclusion

If, after reading this article, you are coming to the conclusion that starting a robotics company might be a tough job, I think you're halfway there. Understanding these basics as a precursor to all the possible, typical, corporate bunk (registering your company, hiring staff, finding a business location, etc.) will give you an accurate mental picture of how high the mountain of work is that you'll have to climb before achieving the kind of success that lets you look back on your trials and tribulations with the pride of a job well done. In the end, building the robot might have been the easy part. **SV**

Author Bio

Timothy J. Brown (Tim) is the Robotics Program Manager for VIA Technologies, Inc. He is also the lead editor and co-founder of the Mobile Robotics website.

VIA Robotics Initiative
www.via.com.tw/en/robotics/robotics.jsp

Mobile Robotics
www.mobilerobotics.org